

IDENTIFICATION

PRODUCT CODE: AH-T???A-DD
DIAGNOSTIC CODE: DFCID
PRODUCT NAME: DFCIDAO CI20 CTP Exerciser
VERSION: 1
DATE RELEASED: September 6, 1985
MAINTAINED BY: Diagnostic Engineering
AUTHOR: Gary Papazian
Gregory A. Scott

COPYRIGHT (C) 1985

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL EQUIPMENT CORPORATION.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE IN EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL EQUIPMENT CORPORATION.

Table of Contents

Page

1.0	OVERVIEW OF DIAGNOSTIC PRODUCT	1
1.1	Scope of this document	1
1.2	General program description	1
1.3	Users and Uses	1
1.4	Prerequisite Hardware	1
1.5	Prerequisite Software	1
2.0	PRODUCT GOALS	3
2.1	Performance Goals	3
2.1.1	Program Run Time	3
2.1.2	Exec/User Mode Capability	3
2.1.3	User Mode Privileges	3
2.1.4	Fault Detection	3
2.1.5	Fault Isolation	3
2.1.6	CPU/Memory Failures	4
2.1.7	Power Fail/Restart	4
2.1.8	General User Interface Philosophy	4
2.2	Compatibility Goals	4
2.2.1	CPU Family	4
2.2.2	Standard Diagnostic Support Software	4
2.2.3	TOPS-10/TOPS-20 Operating systems	4
2.2.4	SPEAR	4
3.0	COMMANDS	6
3.1	CONNECT Command	6
3.2	DISABLE Command	6
3.2.1	DISABLE LOGGING Command	6
3.2.2	DISABLE SPEAR-REPORTING Command	6
3.2.3	DISABLE TRACE Command	6
3.3	ENABLE Command	7
3.3.1	ENABLE LOGGING Command	7
3.3.2	ENABLE SPEAR-REPORTING Command	7
3.3.3	ENABLE TRACE Command	7
3.4	EXIT Command	7
3.5	HELP Command	7
3.6	SHOW Command	8
3.6.1	SHOW ALL Command	8
3.6.2	SHOW LOGGING Command	8
3.6.3	SHOW SPEAR-REPORTING Command	8
3.6.4	SHOW NETWORK Command	8
3.6.5	SHOW NODE Command	8
3.6.6	SHOW TRACE Command	9
3.7	TAKE Command	9
4.0	FUNCTIONAL DESCRIPTION	10
4.1	Remote Console Reservation	10
4.2	Remote Console Operation	10
4.3	Disconnecting From Remote Console	10

1.0 OVERVIEW OF DIAGNOSTIC PRODUCT

1.1 Scope of this document

This document describes the functional definition and design details for the CI Diagnostic/Utilities Protocol Services Program (DFCID).

DFCID is used under TOPS-10 and TOPS-20 to implement the Diagnostic/Utilities Protocol (DUP) for the purpose of communicating with HSC units on the CI.

1.2 General program description

The CI Services Program is a user mode only Program that will run under the TOPS-20 Operating System (Release 6.1 or newer) or TOPS-10 (release 7.03 or newer).

The CI Services Program will enable any terminal accessing the TOPS-10/TOPS-20 system to become the HSC console terminal by "connecting" to it for the purpose of obtaining status and running diagnostics.

Command and Response Packets are passed between the CI Services Program and the HSC by using the Diagnostic/Utilities Protocol (DUP) and the SCS% JSYS (TOPS-20) or the SCS. UUO (TOPS-10).

1.3 Users and Uses

Field Service will use this Program to run remote diagnostic and utility programs on CI nodes.

1.4 Prerequisite Hardware

- o KL10 based system attached to a CI Star Coupler. The system must pass all of the standalone CPU, memory and CI20 diagnostics (including DFPTA and DFCIA).
- o At least one HSC50 system attached to the same CI.

1.5 Prerequisite Software

- o Chronic Operating System running in the HSC50 with the "DUP" module loaded.

- o TOPS-20 Operating System with the SCS% JSYS implemented, or
- o TOPS-10 Operating System with the SCS. UUO implemented

2.0 PRODUCT GOALS

2.1 Performance Goals

To be able to execute the commands received by the user on a terminal and relay messages effectively from the remote program.

2.1.1 Program Run Time

Run time in the traditional sense does not apply. This program is used as a terminal interface between TOPS-10/20 and an HSC50. It requires very little run time during execution.

2.1.2 Exec/User Mode Capability

The CI Services Program will run in User Mode only.

2.1.3 User Mode Privileges

Under TOPS-10, execution of the SCS. UUO is required. This UUO requires the job be running as [1,2] or have the JP.POK priv bit.

Under TOPS-20, execution of the SCS% JSYS is required. The SCS% JSYS requires either Wheel, Operator, or Maintenance capability.

2.1.4 Fault Detection

The CI Services Program will not have any of its own Fault Detection. Any error detected by the remote program will be displayed for the User to see. A JSYS/UUO failure will be displayed on the terminal like most user mode programs. The CI Services program will not do error retries. Hardware errors will be detected and logged by the Operating System.

2.1.5 Fault Isolation

DFCID has no built in isolation capability. Any Functional Fault Detected will be passed back to the user by the remote program for analysis and reporting.

2.1.6 CPU/Memory Failures

DFCID will be aborted by the Operating System in case of memory or CPU failures.

2.1.7 Power Fail/Restart

DFCID will have to be restarted by the user when the system is reloaded.

2.1.8 General User Interface Philosophy

DFCID uses TOPS-20 style Parsing and Recognition. An online help facility is accessed through the HELP command.

2.2 Compatibility Goals

2.2.1 CPU Family

The DFCID is a user program and is Monitor dependent rather than Hardware dependent. It will run on any KL10 Model B 36 bit CPU provided the Monitor is a TOPS-10/20 Monitor that supports the CI.

2.2.2 Standard Diagnostic Support Software

DFCID will not use any of the Standard Diagnostic Support Software. It will be a self contained program with an ".EXE" extension. DFCID does use the system image SYS:GLXLIB.EXE for terminal I/O and command parsing. GLXLIB must be present on any normally functioning TOPS-10/20 system.

2.2.3 TOPS-10/TOPS-20 Operating systems

DFCID will run under TOPS-10 release 7.03 or later and under TOPS-20 release 6.1 or later.

2.2.4 SPEAR

The DFCID will be upward compatible with versions of SPEAR after TOPS-20 release 6.1 and newer. Note: since there is no facility to make Spear entries from a user program under TOPS-10, there is no TOPS-10 Spear functionality.

The following information will be logged into the System Error Log for SPEAR use.

- o The current time when the remote console is reserved.
- o The current time when the remote console is released.
- o Node number.

3.0 COMMANDS

3.1 CONNECT Command

The CONNECT command is used to connect to a node. The node number entered following the CONNECT command. The allowed node numbers are 0 thru 15 (decimal). Any other format will result in an error message.

The CONNECT command connects the user's terminal to a HSC50 and dialogue between the two can now begin.

- o A Control-D typed by the user will disconnect from the remote HSC50 and return the user's terminal to command mode.
- o Two Control-Cs typed while the program is in command or remote mode will always put the user back to monitor level.
- o A Control-Y typed while the program is connected will cause the currently executing HSC50 command to abort. In order to receive another "HSC50>" prompt, you have to type a control-Y after each successfully entered command.

3.2 DISABLE Command

The DISABLE command is used to disable program features.

3.2.1 DISABLE LOGGING Command

The DISABLE LOGGING command closes the log file, which is opened with the ENABLE LOGGING command.

3.2.2 DISABLE SPEAR-REPORTING Command

The DISABLE SPEAR-REPORTING command stops the program from reporting to Spear on TOPS-20 upon each connection and disconnection. The default is to report to Spear. Under TOPS-10 no Spear reporting is done, since there is no facility to make entries into ERROR.SYS.

3.2.3 DISABLE TRACE Command

The DISABLE TRACE command is used to stop the program from displaying progress messages as it is run. Trace mode is normally only used for program debugging, and is off by default.

3.3 ENABLE Command

The ENABLE command is used to set program features.

3.3.1 ENABLE LOGGING Command

The ENABLE LOGGING command is followed by a filename. This file will get a copy of all terminal output. The default filename is DSK:DFCID.LOG. To close the log file, use the EXIT or DISABLE LOGGING commands. Logging is not enabled when the program is started.

3.3.2 ENABLE SPEAR-REPORTING Command

The ENABLE SPEAR-REPORTING command causes the program to report to TOPS-20 Spear upon each connection and disconnection. The default is to report to Spear. Under TOPS-10 no Spear reporting is done, since there is no facility to make entries into ERROR.SYS.

3.3.3 ENABLE TRACE Command

The ENABLE TRACE command is used to stop the program from displaying progress messages as it is run. Trace mode is normally only used for program debugging, and is off by default.

3.4 EXIT Command

The EXIT command is used to stop the program and return to monitor. If there is a log file open, it will be closed before the program exits.

3.5 HELP Command

For general program help, enter just the HELP command, which will cause a message similar to the following to be displayed:

To see all the available commands, type "?" after the program prompt. To see all the available command options, type "?" after the command.

For help on a specific command, type "HELP <command>". For example, HELP CONNECT gives you information about the CONNECT command.

For help on all commands, type 'HELP *'. This will cause the entire help file to be displayed.

To see this text again, type just 'HELP'.

3.6 SHOW Command

The SHOW command is used to display program and CI network status.

3.6.1 SHOW ALL Command

The SHOW ALL command will result in a display of all other SHOW commands except the SHOW NODE command (i.e. SHOW LOGGING, SPEAR-REPORTING, NETWORK, TRACE).

3.6.2 SHOW LOGGING Command

The SHOW LOGGING command will display the name of the log file if open.

3.6.3 SHOW SPEAR-REPORTING Command

The SHOW SPEAR-REPORTING command will display if the program is making reports to Spear or not.

3.6.4 SHOW NETWORK Command

The SHOW NETWORK command will poll all nodes on the CI and display each node number and node type that answers the poll. Example:

```
DFCID> SHOW NETWORK
Node   Type
1.     T-10
15.    HSC
DFCID>
```

3.6.5 SHOW NODE Command

The SHOW NODE command is followed by a node number. It displays returned configuration data for the specified node. Example:

```
DFCID> SHOW NODE (address) 15
Node 15.
```

Virtual circuit state: 3, Open
Maximum datagram size: 264.
Maximum message size: 992.
Software type: HSC
Software version: Y25G
Hardware type: HS50
Node name: HSC015
Port characteristic word: 400000000100

DFCID>

3.6.6 SHOW TRACE Command

The SHOW TRACE command displays the state of TRACE mode. TRACE mode is normally used during program debugging.

3.7 TAKE Command

The TAKE command is followed by a filename containing commands to enter to DFCID's top level command parser. The default filename is DSK:DFCID.CMD.

4.0 FUNCTIONAL DESCRIPTION

4.1 Remote Console Reservation

The DUP server must be explicitly loaded by the HSC50 in order to use DFCID. This is done with the "SET LOAD DUP" command from the HSC50 console. The currently loaded modules may be observed by entering the "SHOW LOAD" command from the HSC50 console.

The DUP server may be in one of three states when the user wishes to reserve the console:

- o The DUP server is loaded and unreserved.
- o The DUP server is loaded and reserved.
- o The DUP server is not loaded.

The user must specify the target HSC with which the user wishes to communicate. The required input to is the node number of the HSC50. The following is an example of the CONNECT command, which is used to connect to the HSC50's console.

```
DFCID> CONNECT (to node) 15
```

To Disconnect and return to command mode, type a single control-D

If the connection is established with the specified node, the following prompt will be displayed on the user's terminal:

```
HSC50>
```

The user can now begin communications with the HSC.

4.2 Remote Console Operation

Once the remote console is reserved, a timer routine will be executed once per 30 seconds. At this time, the program will transmit a GET DUST STATUS message to the HSC to keep the connection. As response data is received from the HSC50, it will be displayed on the user's terminal exactly as it was received.

Enter HSC50 commands after the HSC50> prompt. After each command to the HSC50> prompt, a Control-Y must be typed to get another prompt.

4.3 Disconnecting From Remote Console

To Disconnect and return to command mode, type a single control-D. After the connection is broken, you will return to the DFCID> prompt. A Control-C will also break the connection and return you to the DFCID> prompt.

VERSION: MAINDEC-10-DFCID-A Version 1

AUTHOR: Gary Papazian
 Gregory A. Scott

DATE: September 6, 1985

REASON: Original release.

DOCUMENT

DFCID LST

COPYRIGHT 1985
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

TABLE OF CONTENTS

40	
55	Static Definitions
79	Macros
100	GLXLIB Symbols
4	Essentially GLXINI for -10 and -20 both /GAS
19	User referenced entry vector
47	I%INIT - Initialize the runtime system
77	ACQLIB - Acquire the run-time system
118	CHKLIB - Routine to test if library is in place
148	TRYSEG - Routine to fetch runtime system
61	Assembler Directives
81	Command Parsing Tables
128	GLXLIB Interface
205	General Storage
231	Initial Start Up
327	Top Level Parser
358	CONNECT Command
372	DDT Command
426	HELP Command
575	ENABLE Command
547	DISABLE Command
574	SHOW Command

TABLE OF CONTENTS

736	TAKE Command
832	Quit Command
841	Logging Subroutines
912	Parsing Subroutines
969	Literals
130	Assembler Directives
149	Connect Command
226	Interrupt Channel/Level/Vector Tables
284	Terminal Interrupt Handlers
433	Timeout Interrupt Handler
472	SCS Interrupt Handlers
942	ABORT
990	Disconnect
1016	Get DUST STATUS
1163	Support Subroutines
1446	Show Network Command
1492	Show Node Command
1558	DUP Interface Routines
1801	SCS JSYS/UUO Execution
2191	SCS JSYS/UUO Execution
2319	Spear Reporting Subroutines
2393	Message Buffers

TABLE OF CONTENTS

2439	Datagram Buffers
2465	Storage
2534	Literals

40

Assembler Directives

55

Static Definitions

79

Macros

100

GLXLIB Symbols

4

Essentially GLXINI for -10 and -20 both /GAS

19

User referenced entry vector

47

IXINIT - Initialize the runtime system

77

ACQLIB - Acquire the run-time system

118

CHKLIB - Routine to test if library is in place

148

TRYSEG - Routine to fetch runtime system

61

Assembler Directives

81

Command Parsing Tables

128

GLXLIB Interface

205

General Storage

231

Initial Start Up

327

Top Level Parser

358

CONNECT Command

360 CONNECT command

372

DUP Command

374 DDT command, starts existing DDT or tries to merge it in and start it.

426

HELP Command

428 HELP command

575

ENABLE Command

577 ENABLE Command

584 ENABLE SPEAR

590 ENABLE TRACE

596 ENABLE LOGGING

647

DISABLE Command

649 DISABLE Command

656 DISABLE SPEAR

662 DISABLE TRACE

668 DISABLE LOGGING

674

SHOW Command

676 SHOW Command

683 Show All

692 Show Logging

701 Show Spear

710 Show Trace

719 Show network

724 Show node

736

TAKE Command

738 TAKE Command

832

Quit Command

841

Logging Subroutines

912

Parsing Subroutines

969

Literals

130

Assembler Directives

149

Connect Command

151 CONN - Entered by the 'CONNECT' command from the PARSER

226

Interrupt Channel/Level/Vector Tables

284

Terminal Interrupt Handlers

286 TTYINP - TOPS-20 Terminal input interrupt handler

291 TTYI10 - TOPS-10 Terminal input interrupt handler

296 TT00 - common routine to get input from terminal

389 TTCHRC - common routine to check if something in input buffer

405 CCIN20 - Here on ^C interrupt for TOPS-20

417 CCIN10 - Here on ^C interrupt for TOPS-10

433 *****
Timeout Interrupt Handler :

435 ETIME - TOPS-20 Timer interrupt handler

452 INITIM - Initialize reservation timer

472 *****
SCS Interrupt Handlers

474 DMASRV - TOPS-20 SCS% DMA FINISHED interrupt service routine

479 DGSRV - TOPS-20 SCS% DATAGRAM RECEIVED interrupt service routine

484 EVTSRV - TOPS-20 SCS% EVENT interrupt service routine

489 MSGSRV - TOPS-20 SCS% message available interrupt service routine

494 SCSI10 - TOPS-10 SCS interrupt handler.

509 EVTSO - Common SCS event interrupt handler

631 MSGSRO - Common SCS message received interrupt service routine

735 CONNEC -- Request a connection with a node on the CI.

773 EMULAT - Send EXECUTE LOCAL PROGRAM to run local terminal emulator PSEUDO.

869 SNDHSC - Send a SEND DATA IMMEDIATE packet to the HSC50

942 *****
ABORT

944 ABORTT -- Issue the ABORT DUP command to terminate a program/go back to IDLE

990

Disconnect

992 DISCON -- Disconnect and close the Connection previously made.

1016

GET DUST STATUS

1018 DUSTAT -- Issue GET DUST STATUS to establish characteristics of DUP server

1066 CKDUST - Evaluate the response message returned from the GET DUST STATUS msg

1163

Support Subroutines

1165 CKSTA - Check the status byte of a DUP response message.

1212 EVWAIT -- Wait a specified amount of time for Event Queue Empty

1235 GETTIM - Read current system runtime into S1

1263 MAWAIT -- Wait a specified amount of time for Message Available Interrupt

1289 INIT1 - Initialize buffers, flags, timer and setup terminal

1316 SYSINT - Enable system interrupt channels

1349 RESTOR - Deactivate interrupts and restore terminal to original mode word

1401 CLREFL - Clear event flag table

1409 BUFPTR -- Reset all Buffer Pointers

1446

Show Network Command

1448 SNET - 'SHOW NETWORK' command.

1492

Show Node Command

1494 SNOD - Entered by the 'SHOW NODE n' command

1558

DUP Interface Routines

1560 DUSTST - Get status of DUP Server at remote node.

1598 EXLPGM - Send DUP 'EXECUTE LOCAL PROGRAM' request

1801

SCS JSYS/UUC Execution

1803 SCAINT - Add (set up) Interrupt Channels for SCA related events

1832 SSRCD - Return Connection Data (about a remote node)

1885 SSCON - Request a connection with another node on the CI.

1940 SSEVT - Get Entry from Event Queue

1973 SSDIS - Disconnect from a remote node

2002 SSSMG - Send a message to a remote node.

2037 SSQRM - Queue Message Buffers - receive a message from a DUP server.

2070 SSCSP - Connect State Poll - status of connection

2103 SSSTS - Status Info on Connection (Fast form of .SSCSP)

2132 SSRMG - Receive Message

2165 SSGDE - Get Entry from Data Queue

2191

SCS JSYS/UUO Execution

2193 \$SCS - Do the actual SCS% jsys

2239 FLAGIT - Print error codes from SCS JSYS/UUO.

2319

Spear Reporting Subroutines

2321 SPR1 - Send initial spear report 'diagnostic started'

2363 SPR2 - makes successful completion spear report.

2393

Message Buffers

2395 MESSAGE BUFFER CHAIN - Used only for RECEIVING Messages from the DUP Server

2439

Datagram Buffers

2441 DATAGRAM BUFFER CHAIN - Used only for RECEIVING datagrams from DUP Server.

2465

Storage

2534

Literals

Each DFCID command has an entry in this file. The entries must be alphabetically ordered and must be preceded with an "*" in column 1. Comments are preceded by a "!". Additional entries may be added to this help file, but care must be taken to follow the above.

To see all the available commands, type "?" after the program prompt.
To see all the available command options, type "?" after the command.

For help on a specific command, type 'HELP <command>'.

For help on all commands, type 'HELP *'.

To see this text again, type just 'HELP'.

*CONNECT

The CONNECT command is used to connect to a node. The node number entered following the CONNECT command. The allowed node numbers are 0 thru 15 (decimal). Any other format will result in an error message.

- * A Control-D will disconnect the user's terminal from the remote HSC50 and return the user's terminal to command mode.
- * Two Control-Cs typed while the program is in command or remote mode will always put the user back to monitor level
- * A Control-Y typed while the program is connected will cause the currently executing HSC50 command to abort. In order to receive another 'HSC50>' prompt, you have to type a control-Y after each successfully entered command.

*DISABLE

The DISABLE command is used to disable any of the following features:

LOGGING Close LOG file (default is no log file open).
SPEAR-REPORTING Stop reports to SPEAR (the default).
TRACE Stop showing program progress (the default).

*ENABLE

The ENABLE command is used to set any of the following features:

LOGGING file Start copying all terminal I/O to the specified LOG file (default is not to have a log file).
SPEAR-REPORTING Start spear reporting (default is disabled).
TRACE Show program progress (default is disabled).

*EXIT

The EXIT command is used to stop this program and return to monitor. If there is a log file open, it will be closed before the program exits.

*HELP

The HELP command will list information about the other commands and their options. For example, HELP CONNECT gives you information about

the CONNECT command. To see the options available for each command, type "?" after the command.

SEQ 0027

*QUIT

The QUIT command is used to stop this program and return to monitor. If there is a log file open, it will be closed before the program exits.

*SHOW

The SHOW command is followed by one of the following:

ALL	All show commands except SHOW NODE
LOGGING	Show state of LOGGING switch and log file name
SPEAR-REPORTING	Show state of spear reporting switch
NETWORK	Show all nodes and their types on CI network.
NODE n	Show returned configuration data for node 'n'
TRACE	Show state of TRACE switch

*TAKE

The TAKE command is followed by a filename containing commands to enter as terminal input.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

000001
000012

```
;Z:<GSCOTT.DFCID>DFCIDT.MAC.19 4-Sep-85 14:48:47, Edit by GSCOTT
;Add TTISIZ
;Z:<GSCOTT.DFCID>DFCIDT.MAC.15 3-Sep-85 17:08:32, Edit by GSCOTT
;Add mnemonics for the DUP opcodes
;Z:<GSCOTT.DFCID>DFCIDT.MAC.10 29-Aug-85 19:34:03, Edit by GSCOTT
;Add def of RESTIM, INVCOM
;Z:<GSCOTT.DFCID>DFCIDT.MAC.5 23-Aug-85 19:19:46, Edit by GSCOTT
;Start GLXLIB conversion

DECVER==1          ;Major version
VEDIT==12         ;Edit number: never reset to 0

DEFINE NAME (DECVER,VEDIT),<
UNIVER DFCIDT CI20 DUP Class Driver Symbols Version DECVER('VEDIT')>

NAME \DECVER,\VEDIT^

COMMENT \
        COPYRIGHT (C) 1985
        DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A
SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER
COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO
AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE
SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL EQUIPMENT
CORPORATION.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
EQUIPMENT CORPORATION.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE
USE OR RELIABILITY OF ITS SOFTWARE IN EQUIPMENT WHICH IS NOT
SUPPLIED BY DIGITAL EQUIPMENT CORPORATION.
\
```

```
40          SUBTTL  Assembler Directives
41
42          ;Set version
43
44          000137          LOC      137          ;Get to absolute location .JBVER
45          000137, 000100 000012      <DECVER>B11+VEDIT      ;Set the major version and edit
46          000000          RELOC          ;Back to relocatable code
47
48          ;Assembly parameters
49
50          NOSYM
51          SEARCH  GLXMAC,MONSYM,UUOSYM
52          SALL
53
54          .TEXT  "/RUNAME:DFCID" ;set EXE file name
```

```

55          SUBTTL Static Definitions
56
57          ;Opdefs
58
59          255000 000000          OPDEF NOP [JFCL]          ; no-op
60
61          ;DUP related constants
62
63          000024          TTISIZ==^D5*4          ;Max characters per buffer of data sent
64          072460          RESTIM==^D30000          ;MS between reservation timer ints
65
66          ;DUP opcodes
67
68          000200          INVRES==200          ;DUP Invalid command/opcode issued
69
70          000200          OPCRES==200          ;Opcode offset between req and response
71
72          000001          OPCGDS==1          ;Get DUST status
73          ;          OPCESP==2          ;Execute supplied program
74          000003          OPCELP==3          ;Execute local program
75          000004          OPCSD==4          ;Send data
76          ;          OPCRD==5          ;Receive data
77          000006          OPCABT==6          ;Abort program
78          000007          OPCSDI==7          ;Send data immediate (ASCII text)
  
```

79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

SUBTTL Macros

;Define NAME macro for other modules

DEFINE NAME (MODULE,DECVER,VEDIT,TEXT),<

TITLE MODULE TEXT version DECVER('VEDIT')>

;Define command error macro

DEFINE CMDERR (TEXT),<
JRST [\$CALL CMDLOG
\$TEXT (,<? TEXT>)
JRST TOPLVL]>

;Define macro to output guide words

DEFINE GUIDE (TEXT),<
MOVEI 32,[FLDDB. (.CMNOI,,<Point 7,[ASCIZ/TEXT/]>)]
\$CALL CMDPRS
>

```
100 SUBTTL GLXLIB Symbols
101
102 ;Define symbols that aren't in the GLXMAC on the -20
103
104 000001 .FDSTR==.FDFIL ;Structure containing the file
105 000002 .FDNAM==.FDFIL+1 ;The file name
106 000003 .FDEXT==.FDFIL+2 ;The extension
107 000004 .FDPPN==.FDFIL+3 ;The PPN
108 000005 .FDPAT==.FDFIL+4 ;Start of the path
109
110
111 END
```

NO ERRORS DETECTED

PROGRAM BREAK IS 000000
CPU TIME USED 00:01.367

127P CORE USED

DECVER	10#	16	45		
INVRES	68#				
OPCABT	77#				
OPCELP	74#				
OPCGDS	72#				
OPCRES	70#				
OPCSD	75#				
OPCSDI	78#				
RESTIM	64#				
TTISIZ	63#				
VEDIT	11#	16	45		
.FDEXT	106#				
.FDFIL	104	105	106	107	108
.FDNAM	105#				
.FDPAT	108#				
.FDPPN	107#				
.FDSTR	104#				

SEQ 0033

CMDERR
GUIDE
NAME
NOP

89#
96#
13#
59#

16

83#

SEQ 0034

DFCIDI - Bootstrap code for GALAXY runtime system for DFCIB MACRO %53B(1242) 12:07 6-Sep-85 Page 1
DFCIDI MAC 23-Aug-85 17:42 Essentially GLXINI for -10 and -20 both /GAS

SEQ 0035

```
1      ;Z:<GSCOTT.DFCID>DFCIDI.MAC.2 23-Aug-85 17:42:27, Edit by GSCOTT
2
3      TITLE  DFCIDI - Bootstrap code for GALAXY runtime system for DFCIB
4      SUBTTL  Essentially GLXINI for -10 and -20 both /GAS
5
6      SEARCH GLXMAC,MONSYM,MACSYM,UUOSYM
7      PROLOG(GLXINI,INI)^
8
9
10     000021      INIMAN==:21      ;Maintenance edit number
11     000025      INIDEV==:25      ;Development edit number
12      VERSIN (INI)      ;Generate edit number
13
14     ;Note: Depends on MONTYP being set before call to I%INIT
15     ; MONTYP/ -1 = TOPS-20
16     ; MONTYP/  0 = TOPS-10
17
18     EXTERN  MONTYP
```

```

19      SUBTTL User referenced entry vector
20
21
22      ; All references to the OTS by the user are through ENTVEC table
23      ;
24      000000
25      DEFINE ZZ==0
26      CDO (X),<
27      IFNB <X>,<X::>
28      JRST @.+1
29      EXP ZZ
30      ZZ==ZZ+1
31      >
32      ENTVEC::LIBVEC
33      EXP -1
34
35      WORK: BLOCK 20
36      ARG: BLOCK 6
37
38      ;DEFINE ALL ENTRY POINTS
39      ;TERMINATE VECTOR
40
41      ; Set up correct places to search for the library depending on the system.
42      000637' 44 63 53 00 00 00
43      000640' 63 71 63 00 00 00
44
45      000641' 000000 001036'
46      000642' 000000 001037'
  
```

```

;;INITIALIZE COUNTER
;;MACRO TO DEFINE ENTRY POINTS
;;DEFINE GLOBAL IF NOT NULL
;;DISPATCH
;;STORE INDEX INTO GLXOTS
;;POINT TO NEXT ENTRY
;;END OF CDO MACRO
  
```

```

;Get space for working
;Get space for argument block...
;Note: this space can also be used as
;spillover from WORK
  
```

```

DSKD10: SIXBIT/DSK/
SYSD10: SIXBIT/SYS/

DSKD20: [ASCIZ/DSK:/]
SYSD20: [ASCIZ/SYS:/]
  
```

```

47      SUBTTL I%INIT - Initialize the runtime system
48
49
50      ;Entry point for segment and module initialization
51
52      ;CALL IS:      S1/ Length of IB (Runtime Initialization Block)
53      ;              S2/ Address of IB
54
55      ENTRY  I%INIT
56
57      I%INIT: PUSH  P,S1      ;SAVE INPUT ARGS AWAY
58              PUSH  P,S2      ;
59              PUSHJ P,CHKLIB  ;SEE IF THE LIBRARY IS ALREADY THERE
60              JUMPT INIT.2    ;IT IS - SKIP LOTS OF STUFF
61              PUSHJ P,ACQLIB  ;ACQUIRE THE LIBRARY SYSTEM
62              SKIPE ENTVEC+1   ;Is the entry vector initialized?
63              JRST  INIT.2     ;Yes, don't do it again.
64              SETZM S2        ;Set up index
65
66      INIT.1: ADDM  S1,ENTVEC+1(S2) ;ADJUST INDEX INTO GLXOTS
67              ADDI  S2,2        ;ACCOUNT FOR TWO WORD ENTRIES
68              SKIPL ENTVEC(S2)   ;END OF VECTOR ?
69              JRST  INIT.1      ;NO - LOOP
70
71      INIT.2: POP   P,S2        ;RESTORE USER'S S2
72              POP   P,S1        ;RESTORE USER'S S1
73
74      BEGIN:: HRLI  S1,%GLX     ;LOAD VERSION
75              PUSHJ P,ENTVEC    ;CONTINUE INITIALIZATION
76              $RETT            ;RETURN TO CALLER
  
```

```

77      SUBTTL ACQLIB - Acquire the run-time system
78
79
80      ; Here if the OTS is not in place. If this is a debugging world,
81      ; look first in DSK:. If that fails, or if we are not debugging,
82      ; look in SYS:. If not found, give an error message and quit.
83      ; If found, return the address of the dispatch table for the library
84      ; in S1.
85
86      ; Call:          No arguments
87
88      ; Return:        S1 / Address of OTS dispatch table (returned from TRYSEG)
89
90      000664' 336 00 0 00 000135' ACQLIB: SKIPN DEBUGW ;Debugging ?
91      000665' 254 00 0 00 000673' JRST ACQL.1 ;Nope - skip checking on DSK
92      000666' 332 00 0 00 000000* SKIPE MONTYP ;Skip if TOPS-10
93      000667' 334 01 0 00 000641' SKIPN S1,DSKD20 ;TOPS-20
94      000670' 200 01 0 00 000637' MOVE S1,DSKD10 ;Try generic DSK on -10
95      000671' 260 17 0 00 000733' PUSHJ P,TRYSEG ;Try it again
96      000672' 326 00 0 00 000712' JUMPT ACQL.2 ;If sucessful - finish up
97
98      000673' 332 00 0 00 000666* ACQL.1: SKIPE MONTYP ;Skip if TOPS-10
99      000674' 334 01 0 00 000642' SKIPN S1,SYSD20 ;TOPS-20
100     000675' 200 01 0 00 000640' MOVE S1,SYSD10 ;Try SYS
101     000676' 260 17 0 00 000733' PUSHJ P,TRYSEG ;One more time
102     000677' 326 00 0 00 000712' JUMPT ACQL.2 ;If sucessful - finish up
103
104     000700' 336 00 0 00 000673* SKIPN MONTYP ;Skip if -20
105     000701' 051 03 0 00 001040' OUTSTR [ASCIZ '? GLXINI - Unable to obtain run-time system
106     000702' 332 00 0 00 000700* ;] SKIPE MONTYP ;Skip if -20
107     000703' 047 01 0 00 000012' MONRT. ;Stop
108     000704' 561 01 0 00 001040' HRRORI S1,[ASCIZ '? GLXINI - Unable to obtain run-time system
109     000705' 332 00 0 00 000702* \] SKIPE MONTYP ;Skip if -10
110     000706' 104 00 0 00 000076' PSOUT ;Skip if -10
111     000707' 332 00 0 00 000705* SKIPE MONTYP ;Skip if -10
112     000710' 104 00 0 00 000170' HALTF ;We can continue
113     000711' 254 00 0 00 000673' JRST ACQL.1 ;Return
114
115     000712' 263 17 0 00 000000 ACQL.2: POPJ P,
116
117

```

```

118          SUBTTL CHKLIB - Routine to test if library is in place
119
120          ;CHKLIB looks for the first page of the OTS library. If this page
121          ; is there, it is assumed that the library was loaded with the
122          ; calling program, or has been pre-merged.
123
124          ;CALL IS:      No arguments
125
126          ;TRUE RETURN:  Library is in place
127
128          ;FALSE RETURN: Library is not there
129
130 000713' 332 00 0 00 000707* CHKLIB: SKIPE MONTYP      ;Skip if TOPS-10
131 000714' 254 00 0 00 000722' JRST CHKL20      ;TOPS-20
132 000715' 550 01 0 00 000000* HRRZ S1,JBHRL    ;Get highest addr in high seg
133 000716' 336 00 0 00 000001 SKIPN S1          ;Is there a segment? Yes, skip
134 000717' 634 00 0 00 000000 TDZA TF,TF        ;LOAD "FALSE" INTO TF
135 000720' 561 00 0 00 777777 MOVX TF,TRUE      ;ELSE ITS THERE
136 000721' 263 17 0 00 000000 PCPJ P,          ;RETURN IN EITHER CASE
137
138 000722' 200 01 0 00 000075 CHKL20: MOVE S1,JBHSD    ;FIRST PAGE OF LIBRARY
139 JUMPE S1,[MOVX TF,FALSE ;If zero, then not set up
140 000723' 322 01 0 00 001052' JRST CHKL.1]      ;Return false
141 000724' 505 01 0 00 400000 HRLI S1,.FHSLF    ;FOR THIS PROCESS,
142 000725' 104 00 0 00 000057 RPACS            ;READ THE ACCESS BITS
143 000726' 320 16 0 00 000730' ERJMP .+2        ;IF FAILS, SAY OTS NOT THERE
144 000727' 607 02 0 00 010000 TXNN S2,PA%PEX    ;CHECK FOR PAGE'S EXISTENCE
145 000730' 634 00 0 00 000000 TDZA TF,TF        ;IF PAGES DON'T EXIST, RETURN
146 000731' 561 00 0 00 777777 MOVX TF,TRUE      ;ELSE TRUE,INDICATING OTS IN PLACE
147 000732' 263 17 0 00 000000 CHKL.1: POPJ P,    ;RETURN
  
```

```

148 SUBTTL TRYSEG - Routine to fetch runtime system
149
150 ;CALL IS: S1/ Device to get runtime system from in SIXBIT (TOPS10)
151 ; Address of device string (TOPS20)
152
153 ;TRUE RETURN: Segment is in place
154 ; with S1/ Address of dispatch table
155 ;FALSE RETURN: Segment could not be obtained
156
157 002000 JS.X0==2000 ;EXECUTE-ONLY BIT IN JOB STATUS WORD
158
159 000733' 332 00 0 00 000713* TRYSEG: SKIPE MONTYP ;Skip if TOPS-10
160 000734' 254 00 0 00 000767' JRST TRY20 ;TOPS-20
161
162 000735' 202 01 0 00 000631' MOVEM S1,ARG+0 ;STORE DEVICE NAME
163 000736' 201 01 0 00 000611' MOVEI S1,WORK+0 ;BLT the acs away
164 000737' 251 01 0 00 000630' BLT S1,WORK+17 ;STORE THEM
165 000740' 200 01 0 00 001054' MOVE S1,['GLXLIB'] ;Name of OTS
166 000741' 202 01 0 00 000632' MOVEM S1,ARG+1 ;STORE THE FILE NAME
167 000742' 561 02 0 00 000000 HRROI S2,.GTSTS ;GET JOB STATUS
168 000743' 047 02 0 00 000041 GETTAB S2, ;FROM THE MONITOR
169 000744' 400 02 0 00 000000 SETZ S2, ;CAN'T - ASSUME NOT EXECUTE ONLY
170 000745' 606 02 0 00 002000 TRNN S2,JS.X0 ;EXECUTE ONLY?
171 000746' 634 02 0 00 000002 TDZA S2,S2 ;NO
172 000747' 201 02 0 00 200000 MOVX S2,UU.PHY ;YES - SET FOR PHYSICAL ONLY GETSEG
173 000750' 670 02 0 00 001055' TXO S2,<GETSEG S1,> ;FORM INSTRUCTION
174 000751' 201 01 0 00 000631' MOVEI S1,ARG ;GET LOCATION OF BLOCK
175 000752' 256 00 0 00 000002 XCT S2 ;ATTACH HIGH SEGMENT
176 000753' 634 00 0 00 000000 TDZA TF,TF ;IF FAILS, FAKE A FALSE
177 000754' 561 00 0 00 777777 MOVX TF,TRUE ;ELSE LOAD A "TRUE"
178 000755' 201 01 0 00 001056' MOVE S1,[WORK+S2..S2] ;RESTORE SOME ACS
179 000756' 251 01 0 00 000017 BLT S1,17 ;
180 000757' 336 00 0 00 000000 SKIPN TF, ;True?
181 000760' 263 17 0 00 000000 POPJ P, ;Return
182
183 000761' 200 01 0 00 001057' MOVE S1,[XWD -2..GTUPM] ;Get UPM word for our high seg.
184 000762' 047 01 0 00 000041 GETTAB S1, ;Ask the monitor
185 000763' 505 01 0 00 000000* HRLI S1,.JBHGH ;Monitor won't give it, assume 400000
186 000764' 557 00 0 00 000001 HLRZS S1 ;Get high seg start adr
187 000765' 271 01 0 00 000000* ADDI S1,.JBHDA ;Point past JOB DAT
188
189 000766' 263 17 0 00 000000 POPJ P, ;RETURN

```

190											
191	000767'	200	00	0	00	000001	TRY20: MOVE	TF,S1		;Save device string a second	
192	000770'	201	01	0	00	400000	MOVX	S1,,FHSLF		;For self	
193	000771'	104	00	0	00	000205	GEVEC			;Get entry vector word	
194	000772'	261	17	0	00	000002	PUSH	P,S2		;And save it	
195	000773'	200	01	0	00	000000	MOVE	S1,TF		;Restore s1	
196											
197	000774'	505	01	0	00	440700	HRLI	S1,(POINT 7,0)		;Make the address a byte pointer	
198	000775'	200	02	0	00	001060'	MOVE	S2,[POINT 7,WORK]		;Generate a pointer to destination for	
199										; string	
200	000776'	134	00	0	00	000001	TRY2.1: ILDB	TF,S1		;Get a byte of the device string	
201	000777'	322	00	0	00	001002'	JUMPE	TF,TRY2.2		;Done?	
202	001000'	136	00	0	00	000002	IDPB	TF,S2		;No, save the byte	
203	001001'	254	00	0	00	000776'	JRST	TRY2.1		;Go get another	
204											
205	001002'	200	01	0	00	001064'	TRY2.2: MOVE	S1,[POINT 7,OTSNAM]		;Get the name of the OTS	
206											
207	001003'	134	00	0	00	000001	TRY2.3: ILDB	TF,S1		;Get a byte of the OTS name	
208	001004'	136	00	0	00	000002	IDPB	TF,S2		;Save the byte	
209	001005'	326	00	0	00	001003'	JUMPN	TF,TRY2.3		;Get till done	
210											
211	001006'	205	01	0	00	100001	MOVX	S1,GJ%SH+GJ%OLD		;SHORT GTJFN, OLD FILE	
212	001007'	200	02	0	00	001060'	MOVE	S2,[POINT 7,WORK]		;Get the pointer to the beginning of	
213										; the string	
214	001010'	104	00	0	00	000020	GTJFN			;TRY TO GET A HANDLE ON THE FILE	
215	001011'	254	00	0	00	001033'	JRST	TRY2.4		;FAKE A FALSE RETURN	
216	001012'	660	01	0	00	200000	TXO	S1,GT%ADR		;SET BIT FOR ADDRESS SELECTION	
217	001013'	505	01	0	00	400000	HRLI	S1,,FHSLF		;FOR MYSELF	
218	001014'	200	02	0	00	001065'	MOVE	S2,[1,,760]		;ALL PAGES FROM 1 TO 760	
219	001015'	104	00	0	00	000200	GET			;GET THE SEGMENT	
220	001016'	320	16	0	00	001033'	ERJMP	TRY2.4		;IF GET FAILS, RETURN FALSE	
221											
222	001017'	201	01	0	00	400000	MOVX	S1,,FHSLF		;For self	
223	001020'	104	00	0	00	000205	GEVEC			;Get entry vector word for OTS	
224	001021'	553	02	0	00	000002	HRRZS	S2,S2		;Get only the address	
225	001022'	200	00	0	00	000002	MOVE	TF,S2		;Save it	
226	001023'	242	02	0	00	777767	ADR2PG	S2		;Make a page number	
227	001024'	202	02	0	00	000075	MOVEM	S2,,JBPSO		;Save it for DDT and CHKLIB	
228	001025'	522	02	0	00	000075*	HLLCM	S2,,JBHRL##		;Convince DDT to use the symbols	
229	001026'	262	17	0	00	000002	POP	P,S2		;Restore entry vector word	
230	001027'	104	00	0	00	000204	SEVEC			;Set it back	
231	001030'	200	01	0	00	000000	MOVE	S1,TF		;Get address of dispatch vector	
232	001031'	561	00	0	00	777777	MOVX	TF,TRUE		;We are ok	
233	001032'	254	00	0	00	001035'	JRST	TRY2.5		;go to return	
234											
235	001033'	201	00	0	00	000000	TRY2.4: MOVX	TF,FALSE		;HERE FOR FALSE RETURN	
236	001034'	262	17	0	00	000002	POP	P,S2		;Restore stack	
237	001035'	263	17	0	00	000000	TRY2.5: POPJ	P,		;AND RETURN	

DFCIDI - Bootstrap code for GALAXY runtime system for DFCIB MACRO %53B(1242) 12:07 6-Sep-85 Page 8
DFCIDI MAC 23-Aug-85 17:42 TRYSEG - Routine to fetch runtime system

SEQ 0042

238
239
240 ;Literals
241
242 001036' INI%L:
243 END

NO ERRORS DETECTED

PROGRAM BREAK IS 001066
CPU TIME USED 00:03.632

147P CORE USED

ACQL.1	000673'	I%INIT	000643'	ent	M%IPRC	000244'	int	%OTS	515651	000000	spd
ACQL.2	000712'	I%INT1	000450'	int	M%IPRM	000246'	int	.AOS		000042'	int
ACQLIB	000664'	I%INT2	000452'	int	M%IPSN	000240'	int	.CPUTY		000070'	int
ARG	000631'	I%INT3	000454'	int	M%NXPB	000242'	int	.FHSLF		400000	sin
C%BRVC	000112' int	I%IOFF	000444'	int	M%RELP	000236'	int	.GTSTS		000000	spd
C%CPID	000116' int	I%ION	000442'	int	M%RLNP	000252'	int	.GTUPM		000100	spd
C%INTR	000104' int	I%JINF	000474'	int	M%RMEM	000262'	int	.JBHDA		000765'	ext
C%KPID	000120' int	I%NOW	000436'	int	M%RPAG	000302'	int	.JBHGH		000763'	ext
C%MAXP	000124' int	I%SLP	000446'	int	MONRT.	047040 000012'		.JBHRL		001025'	ext
C%PIDJ	000126' int	I%SOPR	000464'	int	MONTYP	000733'	ext	.JBHSD		000075'	sin
C%RECV	000110' int	I%TIMR	000456'	int	OTSINI	000000'	int	.POPJ		000050'	int
C%REL	000114' int	I%WTO	000466'	int	OUTSTR	051140 000000		.POPJ1		000062'	int
C%RPRM	000102' int	INI%L	001036'		P	000017	spd	.RETE		000040'	int
C%SEND	000106' int	INIDEV	000025	sin	PAXPEX	010000 000000	sin	.RETF		000036'	int
C%SPID	000122' int	INIEDT	000025	sin	PJRST	324740 000000		.RETT		000034'	int
CHKL.1	000732'	INIMAN	000021	sin	PSOUT	104000 000076	int	.SAVE1		000012'	int
CHKL20	000722'	INIT.1	000653'		RPACS	104000 000057	int	.SAVE2		000014'	int
CHKLIB	000713'	INIT.2	000657'		S%CMND	000512'	int	.SAVE3		000016'	int
DEBUGW	000135	JS.XO	002000	spd	S%DATI	000506'	int	.SAVE4		000020'	int
DSKD10	000637'	JUMPF	322000 000000	spd	S%ERR	000520'	int	.SAVE8		000054'	int
DSKD20	000641'	JUMPT	326000 000000	spd	S%EXIT	000524'	int	.SAVET		000022'	int
ENTVEC	000000' int	K%BACK	000554'	int	S%INTR	000522'	int	.SC2UD		000066'	int
ERJMP	320700 000000' int	K%BIN	000552'	int	S%NUMI	000532'	int	.SOS		000044'	int
F%AOBN	000146' int	K%BOUT	000550'	int	S%SCMP	000516'	int	.STKST		000056'	int
F%CHKP	000174' int	K%BUFF	000560'	int	S%SIXB	000510'	int	.STOP		000052'	int
F%DEL	000204' int	K%FLSH	000562'	int	S%TBAD	000526'	int	.SV13		000024'	int
F%DREL	000164' int	K%OPEN	000564'	int	S%TBDL	000530'	int	.SV14		000026'	int
F%FCHN	000206' int	K%RCOC	000534'	int	S%TBLK	000514'	int	.SV15		000030'	int
F%FD	000200' int	K%SOUT	000546'	int	S1	000001	spd	.SV16		000032'	int
F%IBUF	000156' int	K%STYP	000542'	int	S2	000002	spd	.TRSET		000060'	int
F%IBYT	000152' int	K%SUET	000540'	int	SEGIN	104000 000661'	int	.UD2SC		000064'	int
F%INFO	000176' int	K%TPOS	000556'	int	SEVEC	000204'	int	.ZCHNK		000010'	int
F%IOPN	000144' int	K%TXTI	000544'	int	SYSD10	000640'		.ZERO		000046'	int
F%OBUF	000160' int	K%WCOC	000536'	int	SYSD20	000642'		.ZPAGA		000004'	int
F%OBYT	000154' int	L%APOS	000412'	int	T%TEXT	000326'	int	.ZPAGN		000006'	int
F%OOPN	000150' int	L%CBFR	000366'	int	T%TTY	000330'	int				
F%POS	000172' int	L%CENT	000364'	int	TF	000000	spd				
F%REL	000162' int	L%CLST	000360'	int	TRUE	777777 777777'					
F%REN	000202' int	L%CURR	000404'	int	TRYS.1	000776'					
F%REW	000170' int	L%DENT	000370'	int	TRYS.2	001002'					
F%RREL	000166' int	L%DLST	000362'	int	TRYS.3	001003'					
FALSE	000000	L%FIRS	000374'	int	TRYS.4	001033'					
FTJSYS	777777 777777	L%LAST	000376'	int	TRYS.5	001035'					
FTUOS	000000	L%NEXT	000372'	int	TRYS20	000767'					
GET	104000 000200	L%PREM	000402'	int	TRYSEG	000733'					
GETSEG	047000 000040	L%PREV	000400'	int	UU.PHY	200000	spd				
GETTAB	047000 000041	L%RENT	000406'	int	WORK	000611'					
GEVEC	104000 000205	L%SIZE	000410'	int	ZZ	000304	spd				
GJ%OLD	100000 000000	L%STIN.	000000	spd	\$RET	263740 000000	spd				
GJ%SHI	000001 000000	M%ACQP	000234'	int	\$RETF	324740 000036'	spd				
GT%ADR	200000	M%AQNP	000250'	int	\$RETIF	322000 000050'	spd				
GTJFN	104000 000020	M%CLNC	000254'	int	\$RETIT	326000 000050'	spd				
HALTF	104000 000170	M%FPGS	000256'	int	\$RETT	324740 000034'	spd				
I%EXIT	000440' int	M%GMEM	000260'	int	%GLX	000010	spd				
I%HOST	000472' int	M%GPAG	000300'	int	%MOD	475470 515651	spd				

ACQL.1	91	98#	115											
ACQL.2	96	102	117#											
ACQLIB	61	90#												
ARG	36#	162	166	174										
CHKL.1	140	147#												
CHKL20	131	138#												
CHKLIB	59	130#												
DEBUGW	90													
DSKD10	42#	94												
DSKD20	45#	93												
ENTVEC	32#	62	66	68	75									
FALSE	139	235												
GJ%OLD	211													
GJ%SHT	211													
GT%ADR	216													
I%INIT	55	57#												
INI%L	242#													
INIDEV	11#	12												
INIEDT	12#													
INIMAN	10#	12												
INIT.1	66#	69												
INIT.2	60	63	71#											
JS.XO	157#	170												
LSTIN.	8	32												
MONTYP	18#	92	98	104	107	111	113	130	159					
P	8	57	58	59	61	71	72	75	95	101	117	136	147	181
PA%PEX	189	194	229	236	237									
S1	144													
	57	66	72	74	93	94	99	100	109	132	133	138	139	141
	162	163	164	165	166	173	174	178	179	183	184	185	186	187
S2	191	192	195	197	200	205	207	211	216	217	222	231		
	58	64	66	67	68	71	144	167	168	169	170	171	172	173
	175	178	194	198	202	208	212	218	224	225	226	227	228	229
SEGIN	236													
SYSD10	74#													
SYSD20	43#	100												
TF	46#	99												
	134	135	139	145	146	176	177	180	191	195	200	201	202	207
	208	209	225	231	232	235								
TRUE	135	146	177	232										
TRYS.1	200#	203												
TRYS.2	201	205#												
TRYS.3	207#	209												
TRYS.4	215	220	235#											
TRYS.5	233	237#												
TRYS20	160	191#												
TRYSEG	95	101	159#											
UU.PHY	172													
WORK	35#	163	164	178	198	212								
ZZ	24#													
%%.GLX	74													
..MX1	135#	135	136	139#	139	146#	146	147	172#	172	173	177#	177	178
	192#	192	193	211#	211	212	222#	222	223	232#	232	233	235#	235

[illegible]

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

```
;Z:<GSCOTT.DFCID>DFCIDP.MAC.79 4-Sep-85 18:34:18, Edit by GSCOTT
;Intern LOUT%, LOGIFN
;Z:<GSCOTT.DFCID>DFCIDP.MAC.77 4-Sep-85 17:46:25, Edit by GSCOTT
;Correct spelling in SHOW SPEAR/TRACE commands.
;Z:<GSCOTT.DFCID>DFCIDP.MAC.70 30-Aug-85 13:50:21, Edit by GSCOTT
;Intern PCRLF
;Z:<GSCOTT.DFCID>DFCIDP.MAC.68 29-Aug-85 22:20:58, Edit by GSCOTT
;Use IONEOU for output of characters on the 10 (VTDPY on the HSC)
;Z:<GSCOTT.DFCID>DFCIDP.MAC.64 29-Aug-85 21:49:57, Edit by GSCOTT
;Set IO 'JFNs' properly for K%XTI when setting them in CSBTMP.
;Z:<GSCOTT.DFCID>DFCIDP.MAC.62 29-Aug-85 18:57:50, Edit by GSCOTT
;Output CRLF if needed before error message
;Z:<GSCOTT.DFCID>DFCIDP.MAC.60 29-Aug-85 18:44:08, Edit by GSCOTT
;Add TAKE command.
;Z:<GSCOTT.DFCID>DFCIDP.MAC.57 29-Aug-85 18:40:05, Edit by GSCOTT
;Fix logging of failed commands, we called CMDLOG twice and so on
;Z:<GSCOTT.DFCID>DFCIDP.MAC.55 29-Aug-85 17:16:12, Edit by GSCOTT
;Remove DEBUG mode - it doesn't work and its not worth fixing.
;Z:<GSCOTT.DFCID>DFCIDP.MAC.51 29-Aug-85 14:01:38, Edit by GSCOTT
;Add interrupt system base in IB.INT, extern CHNTAB, LEVTAB, PIVECT
;Z:<GSCOTT.DFCID>DFCIDP.MAC.47 27-Aug-85 17:19:32, Edit by GSCOTT
;Add DDT command
;Z:<GSCOTT.DFCID>DFCIDP.MAC.43 27-Aug-85 11:36:56, Edit by GSCOTT
;Remove DISPLAY command, combining SHOW and DISPLAY
;Z:<GSCOTT.DFCID>DFCIDP.MAC.40 26-Aug-85 22:23:50, Edit by GSCOTT
;Repair ENABLE LOGGING.
;Z:<GSCOTT.DFCID>DFCIDP.MAC.39 26-Aug-85 22:08:45, Edit by GSCOTT
;Remove YESNO
;Z:<GSCOTT.DFCID>DFCIDP.MAC.37 26-Aug-85 22:08:27, Edit by GSCOTT
;Remove CONNECT PORT and PORNUM, now we have just CONNECT command.
;Z:<GSCOTT.DFCID>DFCIDP.MAC.34 26-Aug-85 17:18:40, Edit by GSCOTT
;Don't use K%BUFF for IO here, K%FLSH needs to be called too often.
;Z:<GSCOTT.DFCID>DFCIDP.MAC.14 23-Aug-85 16:30:42, Edit by GSCOTT
;Start GLXLIB conversion
```

```
SEARCH DFCIDT
NAME (DFCIDP,\DECVER,\VEDIT,DUP Driver Parse and Dispatch)^
```

COMMENT \

COPYRIGHT (C) 1985

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL EQUIPMENT CORPORATION.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DFCIDP DUP Driver Parse and Dispatch version 1(12)
DFCIDP MAC 4-Sep-85 18:35

MACRO %53B(1242) 12:07 6-Sep-85 Page 1-1

SEQ 0048

56
57
58
59
60

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE
USE OR RELIABILITY OF ITS SOFTWARE IN EQUIPMENT WHICH IS NOT
SUPPLIED BY DIGITAL EQUIPMENT CORPORATION.
\\

```
61          SUBTTL  Assembler Directives
62
63          SEARCH  GLXMAC,MONSYM,UUOSYM,MACSYM
64          PROLOG  (DFCIDP)^
65
66
67          NOSYM
68          SALL
69          .DIREC  FLBLST
70
71          ;Internal symbols
72
73          INTERN  BOUT%%,LOUT%%,PCRLF
74          INTERN  MONTYP,LOGIFN
75
76          ;External symbols
77
78          EXTERN  NODNUM,TRACE,SPEAR,CHNTAB,LEV TAB,PIVECT,TXTARG
79          EXTERN  CONN,SNET,SNOD
80
```



```

128          SUBTTL  GLXLIB Interface
129
130          ;IB - the GLXLIB Interface Block
131
132 000033'   IB:      $BUILD  IB.SZ          ;Here to build the init block
133             $SET      (IB.PRG,,%.MOD)    ;Program name
134             $SET      (IB.OUT,,BOUT%)    ;Default output routine
135             $SET      (IB.FLG,IT.OCT,1)   ;Set open command terminal flag
136             $SET      (IB.FLG,IB.NPF,1)   ;No PFH flag
137             $EOB                      ;End of block
138
139          ;CSB template, BLT'd into place at top level parsing
140
141 000041'   CSBTMP: $BUILD  .CMGJB+1
142             $SET      .CMFLG,RHMASK,REPARS ;Reparse address
143             $SET      .CMBFP,,<POINT 7,BUFFER> ;Pointer to input buffer
144             $SET      .CMPTR,,<POINT 7,BUFFER> ;Start the parse at the beginning
145             $SET      .CMCNT,,BUFSIZ*5-1    ;Size of buffer
146             $SET      .CMABP,,<POINT 7,ATMBUF> ;Pointer to start of atom buffer
147             $SET      .CMABC,,ATMSZ*5-1    ;Size of atom buffer
148             $SET      .CMRTY,,<Point 7,PROMPT> ;Prompt (retry) pointer
149             $SET      .CMGJB,,GJFBLK      ;GTJFN block
150             $EOB
151
152          ;Program prompt
153
154 000053' 104 106 103 111 104 0 PROMPT: BYTE(7)'D','F','C','I','D',76,0 ;DFCIDalglebracket
155
156          ;Fobs and FBs for opening the take file
157
158 000055'   TAKFOB: $BUILD  (FOB.SZ)          ;File open block
159             $SET      (FOB.FD,,TAKFD)      ;File descriptor area
160             $SET      (FOB.CW,FB.BSZ,7)    ;Seven bit bytes
161             $EOB
162
163 000062'   TAKFD:  $BUILD  (FDXSIZ)          ;FD
164             $SET      (.FDLEN,FD.LEN,FDXSIZ) ;Set size of block
165             $EOB
166

```

```

167
168      ;FOBs and FDs for opening the LOG file.
169
170 000156' LOGFOB: $BUILD (FOB.SZ)          ;File open block
171                $SET  (FOB.FD,,LOGFD)      ;File descriptor area
172                $SET  (FOB.CW,FB.BSZ,7)    ;Seven bit bytes
173                $EOB
174
175 000163' LOGFD:  $BUILD (FDXSI)           ;FD size
176                $SET  (FDLEN,FD.LEN,FDXSI) ;Set size of block
177                $EOB
178
179      ;FOBs and FBs for opening the help file
180
181 000257' HFOB20: $BUILD (FOB.SZ)          ;File open block
182                $SET  (FOB.FD,,HLPFD2)     ;File descriptor area
183                $SET  (FOB.CW,FB.BCZ,7)    ;Seven bit bytes
184                $EOB
185
186 000264' HLPFD2: $BUILD (FDSTG+3)         ;FD for TOPS-20
187                $SET  (FDLEN,FD.LEN,FDSTG+3) ;Set size of block
188                $SET  (FDSTG,,<ascii/dsk:d/>) ;First 5 characters of filename
189                $SET  (FDSTG+1,,<ascii/fcid./>) ;Second 5
190                $SET  (FDSTG+2,,<ascii/hlp/>) ;Last 3
191                $EOB
192
193 000270' HFOB10: $BUILD (FOB.SZ)          ;File open block
194                $SET  (FOB.FD,,HLPFD1)     ;File descriptor area
195                $SET  (FOB.CW,FB.BSZ,7)    ;Seven bit bytes
196                $EOB
197
198 000275' HLPFD1: $BUILD (FDPPN+1)         ;FD for TOPS-10
199                $SET  (FDLEN,FD.LEN,FDPPN+1) ;Set size of block
200                $SET  (FDSTR,,<sixbit/DSK/>) ;Structure
201                $SET  (FDNAM,,<sixbit/DFCID/>) ;Filename
202                $SET  (FDEXT,,<sixbit/HLP/>) ;Extension
203                $EOB
204

```

```

205          SUBTTL  General Storage
206
207          ;Program flags and counters, these flags and counters are used internally
208
209 000302'    PDL:   BLOCK   <PLEN==100>          ;The stack
210
211 000402'    MONTYP: BLOCK   1                   ;Monitor type, -1 if TOPS-20
212 000403'    OPSNUM: BLOCK   1                   ;20 if a 20, 10 if a 10
213 000404'    CPUTYP: BLOCK   1                   ;CPU type in sixbit
214 000405'    CPUSER: BLOCK   1                   ;CPU serial number
215 000406'    LOGIFN: BLOCK   1                   ;IFN of log file, 0 if not logging
216 000407'    TAKIFN: BLOCK   1                   ;IFN to take file, 0 if not taking now
217
218          ;Parser storage
219
220 000410'    CSB:     BLOCK   .CMGJB+1             ;The real CSB is here
221 000422'    BUFFER: BLOCK   <BUFSIZ==500>        ;Command buffer
222 001122'    ATMBUF: BLOCK   <ATMSZ==100>         ;Atom buffer
223 001222'    GJFBLK: BLOCK   <GJFSIZ==.GJRTY+2>   ;GTJFN block
224
225          ;Help file reader storage
226
227 001240'    HLPBUF:  BLOCK   ATMSZ                ;Atom buffer for help request
228 001340'    HLPLIN: BLOCK   ^D135/5             ;Line of help text copied here
229 001373'    HLPEOF: BLOCK   1                   ;-1=EOF on help file
230 001374'    HLPIFN: BLOCK   1                   ;IFN of help file
  
```

```

231          SUBTTL Initial Start Up
232
233          ;Determine if TOPS-10 or TOPS-20, and get GLXLIB going
234
235          001375' 200 17 0 00 002450'  SETUP:  MOVE    P,[IOWD PLEN,PDL]      ;Load stack pointer
236          001376' 200 01 0 00 002451'      MOVE    S1,[112,,11]      ;Load %CNMNT monitor type word
237          001377' 047 01 0 00 000041'      GETTAB   S1,              ;This will not call PA1050 on the -20
238          001400' 400 01 0 00 000000'      SETZ     S1,              ;Ops
239          001401' 602 01 0 00 040000'      TXNE     S1,4B23      ;Skip if TOPS-10
240          001402' 334 01 0 00 002452'      SKIPA    S1,[XWD .PRIIN,.PRIOU] ;TOPS-20, set up input/output JFNs
241          001403' 334 01 0 00 002453'      SKIPA    S1,[XWD 377776,377777] ;TOPS-10, set up fake i/o JFNs
242          001404' 334 02 0 00 002454'      SKIPA    S2,[DEC 20]      ;TOPS-20, load a twenty
243          001405' 334 02 0 00 002455'      SKIPA    S2,[DEC 10]      ;TOPS-10, load a ten
244          001406' 334 03 0 00 002456'      SKIPA    T1,[XWD LEVTAB,CHNTAB] ;TOPS-20, interrupt base
245          001407' 334 03 0 00 002457'      SKIPA    T1,[EXP PIVECT]   ;TOPS-10, interrupt base
246          001410' 476 00 0 00 000402'      SETOM    MONTYP      ;TOPS-20, set the flag indicating that
247          001411' 202 01 0 00 000042'      MOVEM    S1,CSBTMP+.CMIOJ   ;Set I/O JFNs properly for -10 or -20
248          001412' 202 01 0 00 000000*     MOVEM    S1,XTARG+.RDIOJ   ;Set I/O JFNs for the K%XTXI
249          001413' 202 02 0 00 000403'      MOVEM    S2,OPSNUM      ;Set ten or twenty
250          001414' 202 03 0 00 000035'      MOVEM    T1,IB+IB.INT    ;Set interrupt base
251
252          ;Get GLXLIB up.
253
254          001415' 201 01 0 00 000006'      MOVEI    S1,IB.SZ      ;Load size of the init block
255          001416' 201 02 0 00 000033'      MOVEI    S2,IB        ; and point to it
256          001417' 260 17 0 00 000000*     $CALL    I%INIT        ;Init the GLXLIB stuff
257
258          ;Get transmit buffer if not already known, release log file IFN if restarted
259
260          repeat 0,<
261              SKIPE   MAPBBA              ;Skip if MAPBBA already there
262              JRST    SETUP3              ;Yes
263              MOVEI   S1,4                 ;Get four pages
264              $CALL   M%AQNP              ;Get a page from the GLXLIB allocator
265              PG2ADR  S1                  ;Get the address from it
266              MOVEM   S1,MAPBBA           ;Save it as transmit buffer
267          >
268          ;end of repeat 0
269
270          ;Close log file if needed upon restart
271
272          001420' 332 01 0 00 000406'  SETUP3: SKIPE   S1,LOGIFN      ;Is there a log file open?
273          001421' 260 17 0 00 000000*  $CALL   F%REL      ;Release the IFN
274          001422' 402 00 0 00 000406'  SETZM   LOGIFN      ;There sure isn't a log file now
275
276          ;Announce ourselves, set up data areas
277
278          001423' 260 17 0 00 001441'      $CALL    CPUNUM      ;Get CPU number, etc.
279          $TEXT    (,<
280          ^I/ANNOUN/^H/[-1]/
281          >
          ;Announce ourselves

```

```

282
283      ;Enable capabilities if TOPS-20
284
285 001426' 336 00 0 00 000402'      SKIPN  MONTYP      ;TOPS-20?
286 001427' 254 00 0 00 001503'      JRST   TOPLVL      ;Nope
287
288 001430' 201 01 0 00 400000      MOVEI   S1,..FHSLF      ;Set up for current process
289 001431' 104 00 0 00 000150      RPCAP%      ;Get my capabilities
290 001432' 606 02 0 00 640000      TRNN     S2,SC%WHL!SC%OPR!SC%MNT ;Any good capabilities present?
291 001433' 260 17 0 00 001424*      $TEXT   (,<% Not enough capabilities to run diagnostic - proceeding>)
292 001435' 201 01 0 00 400000      MOVEI   S1,..FHSLF      ;Incase called above
293 001436' 200 03 0 00 000002      MOVE    T1,S2      ;Enable all capabilities
294 001437' 104 00 0 00 000151      EPCAP%      ;Enable TOPS-20 capabilities
295 001440' 254 00 0 00 001503'      JRST   TOPLVL      ;Enter top level parser
  
```

```

296
297 ;Here to get the CPU number and so on from the monitor.
298
299 001441' 260 17 0 00 000000* CPUNUM: $CALL .CPUTY ;Ask GLXLIB for CPU type
300 001442' 301 01 0 00 000002 CAIL S1,2 ;Is it
301 001443' 303 01 0 00 000005 CAILE S1,5 ; in known range?
302 001444' 634 02 0 00 000002 TDZA S2,S2 ;Nope, say nothing
303 001445' 200 02 0 01 002511' MOVE S2,[EXP 'KI10 ','KL10 ','KS10 ']-2(S1)
304 001446' 202 02 0 00 000404' MOVEM S2,CPUTYP ;Save sixbit CPU type
305
306 001447' 332 00 0 00 000402' SKIPE MONTYP ;Skip if TOPS-10
307 001450' 254 00 0 00 001456' JRST CPUN20 ;Tops-20
308
309 001451' 200 01 0 00 002516' MOVE S1,[20,,11] ;Load %CNSER CPU0 serial number
310 001452' 047 01 0 00 000041 GETTAB S1, ;Get it
311 001453' 400 01 0 00 000000 SETZ S1, ;Error- punt it off
312 001454' 202 01 0 00 000405' MOVEM S1,CPUSER ;Save it there
313 001455' 263 17 0 00 000000 $RET ;Return to caller
314
315 001456' 201 01 0 00 000034 CPUN20: MOVEI S1,.APRID ;Load table for APRID number
316 001457' 104 00 0 00 000010 GETAB% ;Get it
317 001460' 400 01 0 00 000000 SETZ S1, ;Error? punt
318 001461' 202 01 0 00 000405' MOVEM S1,CPUSER ;Save it
319 001462' 263 17 0 00 000000 $RET ;Return
320
321 ;Here is the text string output to terminal and log file when we are started.
322
323 ANNOUN: ITEXT (<DFCID Diagnostic/Utilities Protocol Driver
324 Version ^V/.JBVER##/, ^W/CPUTYP/, TOPS-^D/OPSNUM/, CPU#-^D/CPUSER/
325 >)
326

```

```

327          SUBTTL Top Level Parser
328
329          ;Here on a reparse.
330
331          001501' 200 17 0 00 002450'  REPARS: MOVE    P,[IOWD PLEN,PDL]      ;Here on reparse - reset stack
332          001502' 254 00 0 00 001515'      JRST     PRSCM3          ; and restart parsing
333
334          ;Here for top level commands, reloads the stack and parses a new command.
335
336          001503' 200 17 0 00 002450'  TOPLVL: MOVE    P,[IOWD PLEN,PDL]      ;Point to the stack (again)
337          001504' 200 01 0 00 002537'      MOVE    S1,[XWD CSBTMP,CSB]      ;Initialize the CSB by BLTing
338          001505' 251 01 0 00 000421'      BLT     S1,CSB+.CMGJB      ; the CSB template over it
339
340          ;Normal commands loop.
341
342          001506' 336 01 0 00 000407'  PRSCM1: SKIPN   S1,TAKIFN          ;TAKE command in progress?
343          001507' 254 00 0 00 001513'      JRST     PRSCM2          ;Not in a take command
344          001510' 506 01 0 00 000411'      HRLM     S1,CSB+.CMIOJ      ;Take command, set the input JFN
345          001511' 200 01 0 00 002540'      MOVE    S1,[Point 7,[0]]      ;Point to a zero prompt
346          001512' 202 01 0 00 000412'      MOVEM   S1,CSB+.CMRTY      ;Reset the prompt to be nothing
347
348          001513' 201 02 0 00 002541'  PRSCM2: MOVEI   S2,[FLDDB.(.CMINI)]      ;Load address of init block
349          001514' 260 17 0 00 002347'      $CALL   CMDPRS          ;Init parser
350
351          ;Enter here on reparse (after the CMINI call). Parse a keyword and dispatch.
352
353          001515' 201 02 0 00 002542'  PRSCM3: MOVEI   S2,[FLDDB.(.CMKEY,,MAINCM)] ;Point to top level parsing block
354          001516' 260 17 0 00 002347'      $CALL   CMDPRS          ;parse top level command
355          001517' 550 01 0 02 000000      HRRZ     S1,(S2)          ;Get the keyword address
356          001520' 260 17 0 01 000000      $CALL   (S1)          ;Dispatch
357          001521' 254 00 0 00 001506'      JRST     PRSCM1          ;Get another command
  
```

```

358          SUBTTL  CONNECT Command
359
360          ;*  CONNECT command
361
362          .CONNE: GUIDE  <to node>
363
364          001522' 201 02 0 00 002546'
365          001523' 260 17 0 00 002347'
366          001524' 201 02 0 00 002553'
367          001525' 260 17 0 00 002347'
368          001526' 301 02 0 00 000000
369          001527' 303 02 0 00 000017
370          001530' 254 00 0 00 002571'
371          001531' 202 02 0 00 000000*
          001532' 260 17 0 00 002341'
          001533' 254 00 0 00 000000*
          MOVEI  S2,[FLDDB.(.CMNUM,,^D10,<CI node number>)] ;Point to block
          $CALL  CMDPRS ;Parse that command
          CAIL  S2,0 ;Is it -ive?
          CAILE  S2,^D15 ;Is it greater than 15.?
          CMDERR <Illegal node number ^D/S2/>
          MOVEM  S2,NODNUM ;Save it
          $CALL  CMDCFM ;Parse the confirm
          JRST  CONN ;Do connect
  
```

```

372          SUBTTL  DDT Command
373
374          ;* DDT command, starts existing DDT or tries to merge it in and start it.
375
376          001534' 260 17 0 00 002341' .DDT:  $CALL  CMDCFM          ;Confirm command
377
378          ;Check for DDT loaded by LINK or for VMDDT (TOPS-10)
379
380          001535' 332 01 0 00 000000*      SKIPE  S1,JBDDT          ;DDT present here?
381          001536' 254 00 0 00 001574'      JRST  DDT3          ;Yes, DDT loaded by LINK or VMDDT (T10)
382          001537' 336 00 0 00 000402'      SKIPN  MONTYP        ;TOPS-10?
383          001540' 254 00 0 00 001565'      JRST  DDT1          ;Yep, DDT must not be there
384
385          ;Check for TOPS-20 UDDT, get it as needed
386
387          001541' 332 00 0 00 770000      SKIPE  770000          ;DDT present here?
388          001542' 254 00 0 00 001555'      JRST  DDT6          ;Yes, UDDT is there
389          001543' 205 01 0 00 100001      MOVX   S1,GJ%SH!GJ%OLD    ;Old file short form
390          001544' 561 02 0 00 002575'      HRROI  S2,[ASCIZ/SYS:UDDT.EXE/] ;File to look for
391          001545' 104 00 0 00 000020      GTJFN%          ;Get a JFN for it
392          001546' 320 16 0 00 001562'      ERJMP  DDT4          ;Punt
393          001547' 505 01 0 00 400000      HRLI   S1,.FHSLF      ;Load this fork handle
394          001550' 104 00 0 00 000200      GET%          ;Get the file
395          001551' 320 16 0 00 001557'      ERJMP  DDT5          ;Punt
396          001552' 201 01 0 00 400000      MOVEI  S1,.FHSLF      ;Load this fork handle
397          001553' 200 02 0 00 002600'      MOVE   S2,[JRST SETUP] ;Load the entry vector
398          001554' 104 00 0 00 000204      SEVEC%          ;Set the entry vector
399
400          001555' 201 01 0 00 770000      DDT6:  MOVEI  S1,770000    ;Point to UDDT
401          001556' 254 00 0 00 001574'      JRST  DDT3          ; and off we go
402
403          001557' 201 01 0 01 000000      DDT5:  MOVEI  S1,(S1)      ;Load just the JFN
404          001560' 104 00 0 00 000022'      CLOSFX          ;Release the JFN
405          001561' 320 16 0 00 001562'      ERJMP  .+1          ;Ignore errors
406
407          ;Here if no DDT and I can't load it
408
409          001562' 260 17 0 00 001433*      DDT4:  $TEXT  (,<? DDT not loaded>) ;Nope
410          001564' 263 17 0 00 000000      R:      $RET          ;Return to toplevel
411
412          ;Here to try and merge in VMDDT on TOPS-10
413
414          001565' 202 17 0 00 000401'      DDT1:  MOVEM  P,PDL+PLEN-1    ;Save critical AC that MERGE. smashes
415          001566' 201 01 0 00 002611'      MOVEI  S1,[EXP <SIXBIT/SYS/>,<SIXBIT/VMDDT/>,<SIXBIT/EXE/>,0,0,0]
416          001567' 047 01 0 00 000173'      MERGE. S1,          ;or CALLI ac,173
417          001570' 334 17 0 00 000401'      SKIPA  P,PDL+PLEN-1    ;Owie, load P
418          001571' 334 17 0 00 000401'      SKIPA  P,PDL+PLEN-1    ;OK, start me up
419          001572' 254 00 0 00 001562'      JRST  DDT4          ;Owie
420          001573' 201 01 0 00 700000      MOVEI  S1,700000    ;Point to VMDDT
421
422          ;Here to give message and start DDT
423
424          001574' 260 17 0 00 001562*      DDT3:  $TEXT  (,<To return, type R$G>)
425          001576' 254 00 0 01 000000      JRST  (S1)          ;Enter DDT

```

```

426 SUBTTL HELP Command
427
428 ;* HELP command
429
430 .HELP: MOVEI S2,HLPARS ;Help parse block
431 $CALL CMDPRS ;Parse it
432 $CALL HLPATM ;Copy that atom please
433
434 ;Command has been parsed and thing to get help on is in HLPBUF, set up flags
435
436 HELP1: SETZM HLPEOF ;Not EOF on help file
437 SETZM T4 ;Initially not outputting anything
438 MOVE S1,HLPBUF ;Load help buffer contents
439 CAME S1,[ASCIZ/*/] ;Was it star?
440 SKIPN HLPBUF ;Skip if something typed
441 SETOB T3,T4 ;Yes, output first text you see
442
443 ;Open up the help file for reading.
444
445 DMOVE S1,[EXP <SIXBIT/DSK/>,<ASCII/dsk:d/>] ;Load first place to look
446 $CALL HFIND ;Try and find it there
447 JUMPT HELP2 ;Jump if we got it
448 DMOVE S1,[EXP <SIXBIT/HLP/>,<ASCII/hlp:d/>] ;Load second place
449 $CALL HFIND ;Try and find it there
450 JUMPT HELP2 ;Jump if we got it
451 DMOVE S1,[EXP <SIXBIT/SYS/>,<ASCII/sys:d/>] ;Load third place to look
452 $CALL HFIND ;Try and find it there
453 JUMPF HLPERR ;Jump if we didn't get it

```

```

454
455 ;Ready to read help file, line at a time. First check EOF flag.
456
457 001621' 332 00 0 00 001373' HELP2: SKIPE HLPEOF ;Eof?
458 001622' 254 00 0 00 001645' JRST HELP7 ;Release it and return to top level
459 001623' 260 17 0 00 001652' $CALL HREAD ;Read in a line from the file
460
461 ;Check the first character on the line, if its a '*' then we must check more
462
463 001624' 302 03 0 00 000052' HELP3: CAIE T1,'*' ;Is it a keyword start?
464 001625' 254 00 0 00 001641' JRST HELP6 ;Nope
465 001626' 402 00 0 00 000006' SETZM T4 ;Not outputting text any more
466 001627' 336 00 0 00 001240' SKIPN HLPBUF ;Skip if some specific help request
467 001630' 476 00 0 00 001373' SETOM HLPEOF ;Yes, simulate EOF at first '*'
468 001631' 561 02 0 00 001340' HRROI S2,HLPLIN ;Point to help line
469 001632' 561 01 0 00 001240' HRROI S1,HLPBUF ;Point to buffer
470 001633' 260 17 0 00 000000*$CALL S%SCMP ;Call string comparison routine
471 001634' 200 02 0 00 001240' MOVE S2,HLPBUF ;Load help buffer contents
472 001635' 312 02 0 00 002627' CAME S2,[ASCIZ/*/] ;Was it star?
473 001636' 336 00 0 00 000001' SKIPN S1 ;Exact match?
474 001637' 477 05 0 00 000006' SETOB T3,T4 ;Yes, we should output text now
475 001640' 254 00 0 00 001621' JRST HELP2 ;Get another line
476
477 ;Here if line didn't begin with '*', output the line if output flag is set.
478
479 001641' 561 01 0 00 001340' HELP6: HRROI S1,HLPLIN ;Point to line of text
480 001642' 332 00 0 00 000006' SKIPE T4 ;Outputting now?
481 001643' 260 17 0 00 002270' $CALL SOUT% ;Output to terminal
482 001644' 254 00 0 00 001621' JRST HELP2 ;Loop for more
483
484 ;Here at end of file, determine if something has been printed.
485
486 001645' 336 00 0 00 000005' HELP7: SKIPN T3 ;Skip if something was found
487 001646' 260 17 0 00 001574*$TEXT (,<% Sorry, no information on '^T/HLPBUF/^>')
488 001650' 200 01 0 00 001374' MOVE S1,HLPIFN ;Load the help IFN
489 001651' 324 17 0 00 001421*$PJNST F%REL ;Release it and return

```

```

490
491      ;Here to read a line from the help file into HLPLIN, returns S1/first character
492
493 001652' 200 01 0 00 001374' HREAD: MOVE S1,HLPIFN ;Load the IFN
494 001653' 200 04 0 00 002654'      MOVE T2,[Point 7,HLPLIN] ;Load pointer to line of text
495 001654' 260 17 0 00 000000*      $CALL F%IBYT ;Get the first character of the line
496 001655' 322 00 0 00 001665'      JUMPF HREAD2 ;Punt - EOF
497 001656' 302 02 0 00 000041'      CAIE S2,'!' ;Eat this line?
498 001657' 334 03 0 00 000002'      SKIP T1,S2 ;No, copy the character to T1
499 001660' 474 03 0 00 000000'      SETO T1 ;Indicate it was a comment line
500 001661' 302 02 0 00 000052'      CAIE S2,'*' ;Asterisk?
501 001662' 136 02 0 00 000004'      IDPB S2,T2 ;No, store the 1st character of line
502
503 001663' 260 17 0 00 001654* HREAD1: $CALL F%IBYT ;Get a byte
504 001664' 326 00 0 00 001667'      JUMPT HREAD3 ;Jump if not EOF
505
506 001665' 476 00 0 00 001373' HREAD2: SETOM HLPEOF ;Now it was an EOF
507 001666' 254 00 0 00 001704'      JRST HREAD5 ;Deposit a null and return
508
509 001667' 306 02 0 00 000015' HREAD3: CAIN S2,.CHCRT ;Was it a return?
510 001670' 254 00 0 00 001663'      JRST HREAD1 ;Yes, eat it
511 001671' 306 02 0 00 000012'      CAIN S2,.CHLFD ;Was it a linefeed?
512 001672' 254 00 0 00 001675'      JRST HREAD4 ;Yes, end of line
513 001673' 136 02 0 00 000004'      IDPB S2,T2 ;Store the character
514 001674' 254 00 0 00 001663'      JRST HREAD1 ;Loop
515
516 001675' 321 03 0 00 001652' HREAD4: JUMPL T1,HREAD ;If it was a comment start another line
517 001676' 306 03 0 00 000052'      CAIN T1,'*' ;Line start with a star?
518 001677' 254 00 0 00 001704'      JRST HREAD5 ;Yes
519 001700' 201 02 0 00 000015'      MOVEI S2,.CHCRT ;Load a return
520 001701' 136 02 0 00 000004'      IDPB S2,T2 ; and store it
521 001702' 201 02 0 00 000012'      MOVEI S2,.CHLFD ;Load a line feed
522 001703' 136 02 0 00 000004'      IDPB S2,T2 ; and store that
523
524 001704' 400 02 0 00 000000' HREAD5: SETZ S2 ;Clear S2
525 001705' 136 02 0 00 000004'      IDPB S2,T2 ;Store it here
526 001706' 263 17 0 00 000000'      $RET ;Return

```

```

527
528 ;Parse block for help command.
529
530 001707' 000004 001712' HLPARS: FLDDB. (.CMKEY,,MAINCM,<Help about a command,>,,HAL)
531 001710' 000000 000002'
532 001711' 44 07 0 00 002655'
533 001712' 023005 000000' HAL: FLDDB. (.CMTOK,CM%SDH,<POINT 7,[ASCIZ/*>],<'*' for all of the help file>,,CFM)
534 001713' 44 07 0 00 002627'
535 001714' 44 07 0 00 002662'
536
537 ;Copy atom buffer to help text buffer, returns T1/PDB used
538
539 001715' 200 06 0 00 002670' HLPATM: MOVE T4,[POINT 7,HLPBUF] ;Get pointer to help buffer
540 001716' 403 05 0 00 001240' SETZB T3,HLPBUF ;Make easy check for nothing typed
541 001717' 306 03 0 00 000000' CAIN T1,CFM ;Was it a confirm?
542 001720' 263 17 0 00 000000' $RET ;Yes
543 001721' 306 03 0 00 001712' CAIN T1,HAL ;Was it help all?
544 001722' 254 00 0 00 001731' JRST HLPAT4 ;yes
545 001723' 554 01 0 02 000000' HLRZ S1,(S2) ;Get address of ASCIZ text
546 001724' 505 01 0 00 440700 HRLI S1,(Point 7,) ;Make byte pointer
547
548 001725' 134 03 0 00 000001 HLPAT1: ILDB T1,S1 ;Get first character
549 001726' 136 03 0 00 000006 IDPB T1,T4 ;Not zero, put it in help buffer
550 001727' 326 03 0 00 001725' JUMPB T1,HLPAT1 ;Copy it all
551 001730' 254 00 0 00 002341' JRST CMDCFM ;Confirm that please
552
553 001731' 205 01 0 00 250000 HLPAT4: MOVSJ S1,(<ASCIZ/*>) ;Load a star
554 001732' 202 01 0 00 001240' MOVEM S1,HLPBUF ;Save it as the request type
555 001733' 254 00 0 00 002341' JRST CMDCFM ;Confirm it and return
556
557 ;Here to try and find the help file, called with S1/TOPS-10 dev, S2/TOPS-20 dev
558 ;Returns false if not found, returns true with file open and HLP1FN set.
559
560 001734' 202 01 0 00 000276' HFIND: MOVEM S1,HLPFD1+.FDSTR ;Save TOPS-10 structure name
561 001735' 202 02 0 00 000265' MOVEM S2,HLPFD2+.FDSTG ;Save TOPS-20 first 5 characters
562 001736' 332 00 0 00 000402' SKIPE MONTYP ;Skip if TOPS-10
563 001737' 334 02 0 00 002671' SKIPA S2,[HFOB20] ;TOPS-20
564 001740' 201 02 0 00 000270' MOVEI S2,HFOB10 ;TOPS-10
565 001741' 201 01 0 00 000005' MOVEI S1,FOB.SZ ;Load size of the FOB
566 001742' 260 17 0 00 000000* $CALL FXIOPN ;Open up input file
567 001743' 322000 000000* $RETIF ;Return if false
568 001744' 202 01 0 00 001374' MOVEM S1,HLP1FN ;Save 1FN of help file
569 001745' 324740 000000* $RETT ;Return OK
570
571 ;Here if help file open problem
572
573 001746' 260 17 0 00 001646* HLPERR: $TEXT (,<? Help not available - ^E/S1/>)
574 001750' 263 17 0 00 000000 $RET ;Return

```

```

575          SUBTTL  ENABLE Command
576
577          ;* ENABLE Command
578
579 001751' 201 02 0 00 002704' .ENABL: MOVEI S2,[FLDDB.(.CMKEY,,ENATAB)] ;Point to parse block
580 001752' 260 17 0 00 002347'      $CALL  CMDPRS      ;Parse that command
581 001753' 550 01 0 02 000000      HRRZ    S1,(S2)      ;Set up to go to proper enable routine
582 001754' 254 00 0 01 000000      JRST   (S1)          ;Dispatch
583
584          ;* ENABLE SPEAR
585
586 001755' 260 17 0 00 002341' EN.SPE: $CALL  CMDCFM      ;Parse the confirm
587 001756' 476 00 0 00 000000*      SETOM  SPEAR      ;Enable spear
588 001757' 263 17 0 00 000000      $RET
589
590          ;* ENABLE TRACE
591
592 001760' 260 17 0 00 002341' EN.TRA: $CALL  CMDCFM      ;Parse the confirm
593 001761' 476 00 0 00 000000*      SETOM  TRACE      ;Enable trace
594 001762' 263 17 0 00 000000      $RET

```

```

595
596
597
598 001763' 332 00 0 00 000406'
599 001764' 254 00 0 00 002726'
600 001765' 201 02 0 00 002734'
601 001766' 260 17 0 00 002347'
602 001767' 402 00 0 00 001222'
603 001770' 200 01 0 00 002736'
604 001771' 251 01 0 00 001237'
605 001772' 336 00 0 00 000402'
606 001773' 254 00 0 00 002021'
607
608
609
610 001774' 205 01 0 00 400000'
611 001775' 202 01 0 00 001222'
612 001776' 200 01 0 00 000411'
613 001777' 202 01 0 00 001223'
614 002000' 561 01 0 00 002737'
615 002001' 202 01 0 00 001226'
616 002002' 561 01 0 00 002741'
617 002003' 202 01 0 00 001227'
618 002004' 561 01 0 00 002742'
619 002005' 202 01 0 00 001224'
620 002006' 201 02 0 00 002745'
621 002007' 260 17 0 00 002347'
622 002010' 260 17 0 00 002341'
623 002011' 561 01 0 00 000164'
624
625 002012' 200 03 0 00 002751'
626 002013' 104 00 0 00 000030'
627 002014' 320 16 0 00 002015'
628 002015' 200 01 0 00 000002'
629 002016' 104 00 0 00 000023'
630 002017' 320 16 0 00 002020'
631 002020' 254 00 0 00 002246'
632
633
634
635 002021' 200 01 0 00 002752'
636 002022' 202 01 0 00 001224'
637 002023' 205 01 0 00 435544'
638 002024' 202 01 0 00 001225'
639 002025' 205 01 0 00 446353'
640 002026' 202 01 0 00 001223'
641 002027' 201 02 0 00 002753'
642 002030' 260 17 0 00 002347'
643 002031' 200 01 0 00 002757'
644 002032' 251 01 0 00 002200'
645 002033' 260 17 0 00 002341'
646 002034' 254 00 0 00 002246'

;* ENABLE LOGGING
EN.LOG: SKIPE LOGIFN ;Logging enabled now?
          CMDERR <Logging is already enabled, type DISABLE LOGGING first>
          MOVEI S2,[FLDDB. (.CMNOI,,<Point 7,[ASCIZ/to file/]>)]
          $CALL CMDPRS ;Do it
          SETZM GJFBLK ;Clear first word
          MOVE S1,[GJFBLK,,GJFBLK+1] ;Set up to clear block
          BLT S1,GJFBLK+GJFSIZ-1 ;Clear the block
          SKIPN MONTYP ;Skip if TOPS-20
          JRST EN.L05 ;TOPS-10

;TOPS-20 ENABLE LOGGING command
MOVX S1,GJ%FOU ;File is for output
MOVEM S1,GJFBLK+.GJGEN ; into flags word
MOVE S1,CSB+.CMIOJ ;Load I/O JFNs
MOVEM S1,GJFBLK+.GJSRC ; into block
HRROI S1,[ASCIZ/DFCID/] ;Point at default file name
MOVEM S1,GJFBLK+.GJNAM ;Save for GTJFN
HRROI S1,[ASCIZ/LOG/] ;Default extension
MOVEM S1,GJFBLK+.GJEXT ;Save in GTJFN block
HRROI S1,[ASCIZ/DSK/] ;Get the default structure
MOVEM S1,GJFBLK+.GJDEV ;Save the device
MOVEI S2,[FLDDB. (.CMFIL,,,,<DFCID.LOG>)] ;Output file type
$CALL CMDPRS ;Hello GLXLIB
$CALL CMDCFM ;Confirm that
HRROI S1,LOGFD+.FDSTG ;Point to the FD
MOVX T1,FLD(.JSAOF,JS%DEV)!FLD(.JSAOF,JS%DIR)!FLD(.JSAOF,JS%NAM)!FLD(.JSAOF,JS%TY
P)!FLD(.JSAOF,JS%GEN)!JS%PAF
JFNS% ;JFN to string
ERJMP .+1 ;Error, ignore it
MOVE S1,S2 ;Reload the JFN
RLJFN% ;Release it please
ERJMP .+1 ;Error? Punt it
JRST LOGOPN ;Open the logging file and return

;Here for TOPS-10 ENABLE LOGGING command
EN.L05: MOVE S1,[SIXBIT/DFCID/] ;Get file name
          STORE S1,GJFBLK+.FDNAM ;Save in default block
          MOVS S1,'CMD' ;Get default extension
          STORE S1,GJFBLK+.FDEXT ;Save in block
          MOVS S1,'DSK' ;Get structure name
          STORE S1,GJFBLK+.FDSTR ;Save the structure
          MOVEI S2,[FLDDB. (.CMOFI,,,,<DFCID.LOG>)] ;Input file
          $CALL CMDPRS ;Hello GLXLIB
          MOVE S1,[GJFBLK,,LOGFD] ;Set up to copy into FD
          BLT S1,LOGFD+GJFSIZ-1 ;Clear the block
          $CALL CMDCFM ;Confirm the command
          JRST LOGOPN ;Open the log file and return

```

```

647          SUBTTL  DISABLE Command
648
649          ;* DISABLE Command
650
651 002035' 201 02 0 00 002760' .DISAB: MOVEI  S2,[FLDDB.(.CMKEY,,DISTAB)] ;Point to parse block
652 002036' 260 17 0 00 002347'          $CALL  CMDPRS          ;Parse that command
653 002037' 550 01 0 02 000000          HRRZ  S1,(S2)          ;Set up to go to proper enable routine
654 002040' 254 00 0 01 000000          JRST  (S1)          ;Dispatch
655
656          ;* DISABLE SPEAR
657
658 002041' 260 17 0 00 002341' DA.SPE: $CALL  CMDCFM          ;Parse the confirm
659 002042' 402 00 0 00 001756*          SETZM  SPEAR          ;Disable spear
660 002043' 263 17 0 00 000000          $RET
661
662          ;* DISABLE TRACE
663
664 002044' 260 17 0 00 002341' DA.TRA: $CALL  CMDCFM          ;Parse the confirm
665 002045' 402 00 0 00 001761*          SETZM  TRACE          ;Disable trace
666 002046' 263 17 0 00 000000          $RET
667
668          ;* DISABLE LOGGING
669
670 002047' 336 00 0 00 000406' DA.LOG: SKIPN  LOGIFN          ;Skip if a log file there
671 002050' 254 00 0 00 002774'          CMDERR <Logging was not enabled>
672 002051' 260 17 0 00 002341'          $CALL  CMDCFM          ;Parse the confirm
673 002052' 324 17 0 00 002324'          PJRST  LOGCLS          ;Close the log file
  
```

DFCIDP DUP Driver Parse and Dispatch version 1(12)
DFCIDP MAC 4-Sep-85 18:35 SHOW Command

MACRO %53B(1242) 12:07 6-Sep-85 Page 20

SEQ 0067

```
674 SUBTTL SHOW Command
675
676 ;* SHOW Command
677
678 002053' 201 02 0 00 003000' .SHOW: MOVEI S2,[FLDDB.(.CMKEY,,SHOTAB)] ;Point to parse block
679 002054' 260 17 0 00 002347' $CALL CMDPRS ;Parse that command
680 002055' 550 01 0 02 000000 HRRZ S1,(S2) ;Load dispatch addr
681 002056' 254 00 0 01 000000 JRST (S1) ;Go show it
682
683 ;* Show ALL
684
685 002057' 260 17 0 00 002341' SHOALL: $CALL CMDCFM ;Parse the confirm
686 002060' 260 17 0 00 002066' $CALL SLOG ;Show logging
687 002061' 260 17 0 00 002075' $CALL SSPE ;Show spear
688 002062' 260 17 0 00 002104' $CALL STRA ;Show trace
689 002063' 260 17 0 00 000000* $CALL SNET ;Show network
690 002064' 263 17 0 00 000000 $RET
```

```

691
692      ;* Show Logging
693
694      002065' 260 17 0 00 002341' SHOLOG: $CALL CMDCFM      ;Parse the confirm
695      002066' 336 00 0 00 000406' SLOG:  SKIPN LOGIFN      ;Is logging enabled ?
696      002067' 334 01 0 00 003006'      SKIPN S1,[EXP [ITEXT(disabled)]] ;Disabled
697      002070' 201 01 0 00 003012'      MOVEI S1,[ITEXT(enabled to ^F/LOGFD/)] ;Enabled
698      002071' 260 17 0 00 001746* $TEXT  (,<Program logging is ^I/(S1)/>)
699      002073' 263 17 0 00 000000 $RET
700
701      ;* Show Spear
702
703      002074' 260 17 0 00 002341' SHOSPE: $CALL CMDCFM      ;Parse the confirm
704      002075' 336 00 0 00 002042* SSPE:  SKIPN SPEAR      ;Is spear enabled ?
705      002076' 334 01 0 00 003030'      SKIPN S1,[ASCIZ/dis/] ;Disabled
706      002077' 205 01 0 00 627340      MOVSI S1,(ASCIZ/en/) ;Enabled
707      002100' 260 17 0 00 002071* $TEXT  (,<Program reporting to SPEAR is ^T/S1/abled>)
708      002102' 263 17 0 00 000000 $RET
709
710      ;* Show Trace
711
712      002103' 260 17 0 00 002341' SHOTRA: $CALL CMDCFM      ;Parse the confirm
713      002104' 336 00 0 00 002045* STRA:  SKIPN TRACE      ;Is trace enabled ?
714      002105' 334 01 0 00 003030'      SKIPN S1,[ASCIZ/dis/] ;Disabled
715      002106' 205 01 0 00 627340      MOVSI S1,(ASCIZ/en/) ;Enabled
716      002107' 260 17 0 00 002100* $TEXT  (,<Program trace mode is ^T/S1/abled>)
717      002111' 263 17 0 00 000000 $RET
718
719      ;* Show network
720
721      002112' 260 17 0 00 002341' SHONET: $CALL CMDCFM      ;Parse the confirm
722      002113' 254 00 0 00 002063* JRST  SNET      ;Do the work
723
724      ;* Show node
725
726      002114' 201 02 0 00 003065' SHONOD: GUIDE <address> ;node
727      002115' 260 17 0 00 002347'      MOVEI S2,[FLDDB.(.CMNUM,,^D10,<CI node number>)] ;Point to block
728      002116' 201 02 0 00 002553'      $CALL CMDPRS      ;Parse that command
729      002117' 260 17 0 00 002347'      CAIL S2,0 ;Is it -ive?
730      002120' 301 02 0 00 000000      CAILE S2,^D15 ;Is it greater than 15.?
731      002121' 303 02 0 00 000017      CMDERR <Illegal node number ^D/S2/>
732      002122' 254 00 0 00 002571'      MOVEM S2,NODNUM ;Save it
733      002123' 202 02 0 00 001531* $CALL CMDCFM      ;Parse the confirm
734      002124' 260 17 0 00 002341' JRST  SNOD      ;Do work and return
735      002125' 254 00 0 00 000000*

```

```

736 SUBTTL TAKE Command
737
738 ;* TAKE Command
739
740 .TAKE: SKIPE TAKIFN ;In a TAKE already?
741 CMDERR <Nested TAKE files are illegal>
742 SETZM GJFBLK ;Clear first word
743 MOVE S1,[GJFBLK,,GJFBLK+1] ;Set up to clear block
744 BLT S1,GJFBLK+GJFSIZ-1 ;Clear the block
745 MOVEI S2,[FLDDB.(.CMNOI,,<Point 7,[ASCIZ/commands from/]>)]
746 $CALL CMDPRS ;Do the noise
747 SKIPN MONTYP ;Skip if TOPS-20
748 JRST TAKE1 ;TOPS-10
749
750 ;TOPS-20 take command
751
752 MOVX S1,GJ%OLD ;File must exist
753 MOVEM S1,GJFBLK+.GJGEN ; into flags word
754 MOVE S1,CSB+.CMIOJ ;Load I/O JFNs
755 MOVEM S1,GJFBLK+.GJSRC ; into block
756 HRROI S1,[ASCIZ/DFCID/] ;Point at default file name
757 MOVEM S1,GJFBLK+.GJNAM ;Save for GTJFN
758 HRROI S1,[ASCIZ/CMD/] ;Default extension
759 MOVEM S1,GJFBLK+.GJEXT ;Save in GTJFN block
760 HRROI S1,[ASCIZ/DSK/] ;Get the default structure
761 MOVEM S1,GJFBLK+.GJDEV ;Save the device
762 MOVEI S2,[FLDDB.(.CMFIL,,,,<DFCID.CMD>)] ;Input file type
763 $CALL CMDPRS ;Hello GLXLIB
764 $CALL CMDCFM ;Confirm that
765 HRROI S1,TAKFD+.FDSTG ;Point to the FD
766 MOVEM S2,TAKIFN ;Save the take JFN
767 SETZ T1, ;Default format
768 JFNS% ;JFN to string
769 ERJMP .+1 ;Error, ignore it
770 MOVE S1,TAKIFN ;Get the JFN again
771 MOVX S2,OF%RD!FLD(7,OF%BSZ) ;Read 7 bit bytes
772 OPENF% ;Pry it open
773 ERJMP TAKE6 ;Error, punt
774 JRST TAKE3 ;Give startip message
  
```

```

775
776 ;Here for TOPS-10 TAKE command
777
778 002166' 200 01 0 00 002752' TAKE1: MOVE S1,[SIXBIT/DFCID/] ;Get file name
779 002167' 202 01 0 00 001224' STORE S1,GJFBLK+.FDNAM ;Save in default block
780 002170' 205 01 0 00 435544' MOVSJ S1,'CMD' ;Get default extension
781 002171' 202 01 0 00 001225' STORE S1,GJFBLK+.FDEXT ;Save in block
782 002172' 205 01 0 00 446353' MOVSJ S1,'DSK' ;Get structure name
783 002173' 202 01 0 00 001223' STORE S1,GJFBLK+.FDSTR ;Save the structure
784 002174' 201 02 0 00 003123' MOVEI S2,[FLDDB. (.CMIFI,...,<DFCID.CMD>)] ;Input file
785 002175' 260 17 0 00 002347' $CALL CMDPRS ;Hello GLXLIB
786 002176' 200 01 0 00 003127' MOVE S1,[GJFBLK.,TAKFD] ;Set up to copy into FD
787 002177' 251 01 0 00 000077' BLT S1,TAKFD+GJFSIZ-1 ;Clear the block
788 002200' 260 17 0 00 002341' $CALL CMDCFM ;Confirm the command
789 002201' 201 01 0 00 000005' MOVEI S1,FOB.SZ ;Load size of FOB
790 002202' 201 02 0 00 000055' MOVEI S2,TAKFOB ;Point to the FOB
791 002203' 260 17 0 00 001742* $CALL F%IOPN ;Open up the thing
792 002204' 322 00 0 00 002211' JUMPF TAKE6 ;Punt if an error
793 002205' 202 01 0 00 000407' MOVEM S1,TAKIFN ;Save the IFN
794
795 ;Here to give startup message and then return for commands.
796
797 002206' 260 17 0 00 002107* TAKE3: $TEXT (,<[Processing ^F/TAKFD/]>)
798 002210' 263 17 0 00 000000 $RET ;Return to get commands from file
799
800 ;Here if an error opening the file, etc.
801
802 002211' 260 17 0 00 002206* TAKE6: $TEXT (,<? ^E/S1/>) ;Give error message
803 002213' 254 00 0 00 002214' JRST TAKABT ;Abort the take

```

```

804
805 ;Here to abort a TAKE file, returns to TOPLVL
806
807 002214' 336 01 0 00 000407' TAKABT: SKIPN S1,TAKIFN ;Skip if an IFN there
808 002215' 254 00 0 00 001503' JRST TOPLVL ;None there, restart now
809 002216' 332 00 0 00 000402' SKIPE MONTYP ;Skip if TOPS-10
810 002217' 104 00 0 00 000022' CLOSFX ;TOPS-20, close the file
811 002220' 320 16 0 00 002221' ERJMP .+1 ;Ignore errors
812 002221' 336 00 0 00 000402' SKIPN MONTYP ;Skip if TOPS-20
813 002222' 260 17 0 00 001651* $CALL F%REL ;TOPS-10, release the file
814 002223' 402 00 0 00 000407' SETZM TAKIFN ;No longer an IFN/JFN
815 002224' 254 00 0 00 001503' JRST TOPLVL ;Restart the parsing
816
817 ;Here to check if we had an EOF on the take file.
818
819 002225' 334 00 0 00 000402' TAKEOF: SKIPN MONTYP ;Skip if TOPS-20
820 002226' 254 00 0 00 002235' JRST TAKE01 ;TOPS-10
821 002227' 336 01 0 00 000407' SKIPN S1,TAKIFN ;Get input file JFN for take file
822 002230' 324740 000000* $RETF ;No take file so not EOF
823 002231' 104 00 0 00 000024 GTSTS% ;Get the file's status
824 002232' 607 02 0 00 001000 TXNN S2,GS%EOF ;At end of file?
825 002233' 324740 002230* $RETF ;Not at end of file, return false
826 002234' 324740 001745* $RETT ;At end of file, return true
827
828 002235' 332 00 0 00 000407' TAKE01: SKIPE TAKIFN ;Skip if no take file
829 002236' 302 01 0 00 000001 CAIE S1,EREOF$ ;End of file?
830 002237' 324740 002233* $RETF ;No take file or not EOF
831 002240' 324740 002234* $RETT ;Yes, take file and EOF

```

```
832 SUBTTL Quit Command
833
834 ;Here for quit command.
835
836 002241' 201 02 0 00 003155' .QUIT: MOVEI S2,[FLDDB. (.CMNOI,,<Point 7,[ASCIIZ/to monitor/]>)]
837 002242' 260 17 0 00 002347' $CALL CMDPRS ;Output the noise words
838 002243' 260 17 0 00 002341' $CALL CMDCFM ;Confirm the diagnostic exit
839 002244' 260 17 0 00 002324' $CALL LOGCLS ;Disable logging
840 002245' 254 00 0 00 000000* JRST I%EXIT ;Exit program
```

```

841 SUBTTL Logging Subroutines
842
843 ;LOGOPN - Subroutine to open the log file, LOGFB already set up.
844
845 002246' 201 01 0 00 000005 LOGOPN: MOVEI S1,FOB.SZ ;Load size of FOB
846 002247' 201 02 0 00 000156' MOVEI S2,LOGFOB ;Point to the FOB
847 002250' 260 17 0 00 000000* $CALL F%AOBN ;Open up the thing append mode
848 002251' 322 00 0 00 002260' JUMPF LOGOP2 ;Error.. punt it
849 002252' 202 01 0 00 000406' MOVEM S1,LOGIFN ;Save the IFN
850 $TEXT (LOUT%%,<^I/ANNOUN/^F/LOGFD/ opened at ^H/[-1]/
851 002253' 260 17 0 00 002211* >) ;Start the log file
852 002255' 260 17 0 00 002253* $TEXT (T%TTY,<[^F/LOGFD/ opened]>) ;Output logging filename
853 002257' 263 17 0 00 000000 $RET ;Return
854
855 002260' 260 17 0 00 002255* LOGOP2: $TEXT (,<? ^E/S1/, logging not enabled>)
856 002262' 263 17 0 00 000000 $RET ;Give message and return
857
858 ;PCRLF - output CRLF to terminal
859
860 002263' 304 00 0 00 000000 PCRLF: $SAVE <S1> ;Save an AC
861 HRROI S1,[ASCIZ/
862 002267' 561 01 0 00 002460' /] ;Fall through to SOUT%%
863
864 ;SOUT%% - Routine that is called for string output to terminal.
865 ; S1/ address of ASCIZ string
866
867 002270' 260 17 0 00 000000* SOUT%%: $SAVE <P1,P2> ;Save an AC
868 002271' 201 07 0 01 000000 MOVEI P1,(S1) ;Load the address of the string
869 002272' 505 07 0 00 440700 HRLI P1,(Point 7) ;Make a byte pointer out of it
870
871 002273' 13 01 0 00 000007 SOUT%1: ILDB S1,P1 ;Load a byte
872 002274' 322 01 0 00 C01743* JUMPE S1,.POPJ ;Jump if done
873 002275' 260 17 0 00 002277' $CALL BOUT%% ;Output the byte to the file
874 002276' 254 00 0 00 002273' JRST SOUT%1 ;Loop for all bytes in the string

```

```

875
876 ;BOUT%% - Routine that is called for printing a character in S1.
877 ; Sends character to terminal and log file if opened.
878
879 002277' 332 00 0 00 000406' BOUT%%: SKIPE LOGIFN ;Skip if no logging IFN
880 002300' 260 17 0 00 002307' $CALL LOUT%% ;Output to log file
881 002301' 332 00 0 00 000402' SKIPE MONTYP ;Skip if TOPS-10
882 002302' 254 00 0 00 002305' JRST BOUT%2 ;TOPS-20
883 002303' 051 15 0 00 000001' IONEOU S1 ;Image ONE Output from S1
884 002304' 263 17 0 00 000000' $RET ;Return
885 002305' 104 00 0 00 000074' BOUT%2: PBOUT% ;Primary Byte OUTput from S1
886 002306' 263 17 0 00 000000' $RET ;Return
887
888 ;LOUT%% - Routine to output character to LOG file, S1/ character
889
890 002307' 304 00 0 00 000000' LOUT%%: $SAVE <S1,S2> ;Save the ACs in question
891 002314' 200 02 0 00 000001' MOVE S2,S1 ;Copy byte to S2
892 002315' 336 01 0 00 000406' SKIPN S1,LOGIFN ;Load the IFN
893 002316' 260 17 0 00 000000* $STOP (LOC,Output to Log file when not opened)
894 002320' 260 17 0 00 000000* $CALL F%OBYT ;Output the byte
895 002321' 326000 002274* $RETI ;Return if OK
896 002322' 260 17 0 00 002260* $TEXT (I%TTY,<? Output error to LOG file: ^E/S1/>)
897 ;Fall thru to close LOG file
898
899 ;LOGCLS - Subroutine to close the log file, checks if it was open first
900
901 002324' 336 01 0 00 000406' LOGCLS: SKIPN S1,LOGIFN ;Skip and load existing IFN
902 002325' 263 17 0 00 000000' $RET ;Return now
903 $TEXT (LOUT%<
904 002326' 260 17 0 00 002322* ^F/LOGFD/ closed at ^H/[-1]/>)
905 002330' 260 17 0 00 002222* $CALL F%REL ;Release the IFN
906 002331' 402 00 0 00 000406' SETZM LOGIFN ;Clear that IFN
907 002332' 336 00 0 00 000000' SKIPT ;Skip if it went well
908 002333' 260 17 0 00 002326* $TEXT (,<? ^F/LOGFD/ close error: ^E/S1/>)
909 002335' 332 00 0 00 000000' SKIPF ;Skip if it didn't go so well
910 002336' 260 17 0 00 002333* $TEXT (,<[^F/LOGFD/ closed]>) ;Close message
911 002340' 263 17 0 00 000000' $RET ;Return OK

```

```

912          SUBTTL Parsing Subroutines
913
914          ;CMDCFM - subroutine to parse a confirm.
915
916          002341' 304 00 0 00 000000'   CMDCFM: $SAVE    <S1,S2>           ;Save 2 ACs
917          002346' 201 02 0 00 000000'   MOVEI    S2,CFM           ;Just a confirm please
918                                           ;Fall thru to parse it
919
920          ;CMDPRS - Come here to parse a command with function block in S2.
921          ;Returns S2/result, T1/PDB used, S1/flags
922
923          002347' 201 01 0 00 000410'   CMDPRS: MOVEI    S1,CSB           ;Point to the CSB
924          002350' 260 17 0 00 000000*'   $CALL    S%CMND           ;Hello GLXLIB
925          002351' 322 00 0 00 002370'   JUMPF    PARERR           ;Pass along a false return
926          002352' 200 01 0 02 000000'   MOVE     S1,CR.FLG(S2)      ;Get flag word of reply block
927          002353' 603 01 0 00 200000'   TXNE     S1,CM%NOP         ;Was no-parse set?
928          002354' 254 00 0 00 002370'   JRST     PARERR           ;Yep
929          002355' 550 03 0 02 000002'   HRRZ     T1,CR.PDB(S2)     ;Load the PDB used
930          002356' 200 02 0 02 000001'   MOVE     S2,CR.RES(S2)     ;Load the result
931          002357' 302 03 0 00 000000'   CAIE     T1,CFM           ;Was it the confirm one
932          002360' 263 17 0 00 000000'   $RET                                           ;No, return to caller
933
934          ;CMDLOG - log a the contents of the command parse buffer
935
936          002361' 332 00 0 00 000406'   CMDLOG: SKIPE   LOGIFN           ;Skip if not logging
937          002362' 260 17 0 00 002336*'   $TEXT    (LOUT%,<^T/PROMPT/^T/BUFFER/^A>)
938          002364' 332 00 0 00 000407'   SKIPE     TAKIFN           ;Skip if not taking
939          002365' 260 17 0 00 002362*'   $TEXT    (T%TTY,<^T/PROMPT/^T/BUFFER/^A>)
940          002367' 263 17 0 00 000000'   $RET                                           ;Return
941
942          ;PARERR - Come here on parse error to print standard message and return.
943
944          002370' 260 17 0 00 002225'   PARERR: $CALL    TAKEOF           ;Check for take command file EOF
945          002371' 322 00 0 00 002377'   JUMPF    PARERR1          ;Jump if not EOF
946          002372' 336 00 0 00 000407'   SKIPN    TAKIFN           ;Skip if a TAKE IFN there
947          002373' 254 00 0 00 002377'   JRST     PARERR1          ;Nope
948          $TEXT    (<
949          002374' 260 17 0 00 002365*'   [End of ^F/TAKFD/]>)           ;End of take file
950          002376' 254 00 0 00 002214'   JRST     TAKABT           ;Abort take file and return
951
952          002377' 260 17 0 00 000000*'   PARERR1: $CALL    S%ERR           ;Load error text pointer to S1
953          002400' 336 00 0 00 000000'   SKIPT                                         ;Skip if it was OK
954          002401' 201 01 0 00 003360'   MOVEI    S1,[ASCIIZ/Command error/] ;No? Load a default message
955          002402' 260 17 0 00 002361'   $CALL    CMDLOG           ;Log the command
956
957          002403' 200 02 0 00 000414'   PARERR3: MOVE     S2,CSB+.CMPTR      ;Load pointer into buffer
958          002404' 134 03 0 00 000002'   ILDB     T1,S2           ;Load a byte
959          002405' 306 03 0 00 000015'   CAIN     T1,.CHCRT        ;Return?
960          002406' 254 00 0 00 002411'   JRST     PARERR4          ;Yep
961          002407' 326 03 0 00 002404'   JUMPN    T1,PARERR3       ;Jump if not a null
962          002410' 260 17 0 00 002263'   $CALL    PCRLF           ;Output CRLF
963
964          002411' 260 17 0 00 002374*'   PARERR4: $TEXT    (<? ^T/(S1)/, type HELP for help>) ;Publish message
965          002413' 336 00 0 00 000407'   SKIPN    TAKIFN           ;Skip if in a take
966          002414' 254 00 0 00 001503'   JRST     TOPLVL          ;No, restart everything
  
```

DFCIDP DUP Driver Parse and Dispatch version 1(12) MACRO %53B(1242) 12:07 6-Sep-85 Page 28-1
DFCIDP MAC 4-Sep-85 18:35 Parsing Subroutines

SEQ 0076

967 002415' 260 17 0 00 002411* \$TEXT (, <? Aborting TAKE file ^F/TAKFD/>) ;Give me a sign, o Leader
968 002417' 254 00 0 00 002214' JRST TAKABT ;Abort it and return

DFCIDP DUP Driver Parse and Dispatch version 1(12) MACRO %53B(1242) 12:07 6-Sep-85 Page 29
DFCIDP MAC 4-Sep-85 18:35 Literals

SEQ 0077

```
969          SUBTTL Literals
970
971          ;Literals dumped here
972
973 002420'    LIT..P: XLIST          ;LIT
974            LIST
975
976            001375'          END    SETUP
```

NO ERRORS DETECTED

PROGRAM BREAK IS 003431
CPU TIME USED 00:45.463

154P CORE USED

SEQ 0078

[illegible]

FDXSIZ	163	164	175	176										
FOB.CW	160	172	183	195										
FOB.FD	159	171	182	194										
FOB.SZ	158	170	181	193	565	789	845							
FWMASK	133	134	143	144	145	146	147	148	149	159	171	182	188	189
	190	194	200	201	202									
GJ%FOU	610													
GJ%OLD	389	752												
GJ%SHT	389													
GJFBLK	149	223#	602	603	604	611	613	615	617	619	636	638	640	643
	742	742	744	753	755	757	759	761	779	781	783	786		
GJFSIZ	223#	604	644	744	787									
GS%EOF	824													
HAL	530	533#	543											
HELP1	436#													
HELP2	447	450	457#	475	482									
HELP3	463#													
HELP6	464	479#												
HELP7	458	486#												
HFIND	446	449	452	560#										
HFOB10	193#	564												
HFOB20	181#	563												
HLPARS	430	530#												
HLPAT1	548#	550												
HLPAT4	544	553#												
HLPATM	432	539#												
HLPBUF	227#	438	440	466	469	471	539	540	554					
HLPEOF	229#	436	457	467	506									
HLPERR	453	573#												
HLPFD1	194	198#	560											
HLPFD2	182	186#	561											
HLPIFN	230#	488	493	568										
HLPLIN	228#	468	479	494										
HREAD	459	493#	516											
HREAD1	503#	510	514											
HREAD2	496	506#												
HREAD3	504	509#												
HREAD4	512	516#												
HREAD5	507	518	524#											
IXEXIT	840													
IXINIT	256													
IB	132#	250	255											
IB.FLG	135	136												
IB.INT	250													
IB.NPF	136													
IB.OUT	134													
IB.PRQ	133													
IB.SZ	132	254												
IT.OCT	135													
JS%DEV	625													
JS%DIR	625													
JS%GEN	625													
JS%NAM	625													

SEQ 0079

JS%PAF	625													
JS%TYP	625													
LEV TAB	78#	244												
LIT...P	973#													
LOGCLS	673	839	901#											
LOGFD	171	175#	623	643	644									
LOGFOB	170#	846												
LOGIFN	74	215#	271	273	598	670	695	849	879	892	901	906	936	
LOGOP2	848	855#												
LOGOPN	631	646	845#											
LOUT%	73	880	890#											
LSTIN.	65	137	138	138#	150	151	151#	161	162	162#	165	166	166#	173
	174	174#	177	178	178#	184	185	185#	191	192	192#	196	197	197#
	203	204	204#	281	292	325	368	410	425	488	574	599	671	696
	697	699	708	717	732	741	798	803	852	853	856	861	861#	868
	891	891#	894	894#	897	905	909	911	917	917#	938	940	950	965
	968													
MAINCM	90#	353	531											
MONTYP	74	211#	246	285	306	382	562	605	747	809	812	819	881	
NODNUM	78#	369	733											
OF%BSZ	771													
OF%RD	771													
OPSNUM	212#	249												
P	65	235	331	336	414	417	418	861	891	917				
P1	860	867	868	869	871	890	916							
P2	860	867	890	916										
P3	860	867	890	916										
P4	860	867	890	916										
PARER1	945	947	952#											
PARER3	958#	961												
PARER4	960	964#												
PARERR	925	928	944#											
PCRLF	73	860#	962											
PDL	209#	235	331	336	414	417	418							
PIVECT	78#	245												
PLEN	209#	235	331	336	414	417	418							
PROMPT	148	154#												
PRSCM1	342#	357												
PRSCM2	343	348#												
PRSCM3	332	353#												
R	410#													
REPARS	142	331#												
RHMASK	142													
S%CMND	924													
S%ERR	952													
S%SCMP	470													
S...LOC	893#													
Si	236	237	238	239	240	241	247	248	254	271	288	292	300	301
	303	309	310	311	312	315	317	318	337	338	342	344	345	346
	355	356	380	389	393	396	400	403	415	416	420	425	438	439
	445	448	451	469	473	479	488	493	545	546	548	553	554	560
	565	568	581	582	603	604	610	611	612	613	614	615	616	617
	618	619	623	628	635	636	637	638	639	640	643	644	653	654

S2	680	681	696	697	705	706	714	715	743	744	752	753	754	755
	756	757	758	759	760	761	765	770	778	779	780	781	782	783
	786	787	789	793	807	821	829	845	849	860	861	868	871	872
	883	890	891	892	901	916	923	926	927	954				
	242	243	249	255	290	293	302	303	304	348	353	355	362	364
	366	367	369	390	397	430	468	471	472	497	498	500	501	509
	511	513	519	520	521	522	524	525	545	561	563	564	579	581
	600	620	628	641	651	653	678	680	726	728	730	731	733	745
	762	766	771	784	790	824	836	846	890	891	916	917	926	929
	930	957	958											
SC%MNT	290													
SC%OPR	290													
SC%WHL	290													
SETUP	235#	397	976											
SETUP3	271#													
SHOALL	121	685#												
SHOLOG	122	694#												
SHONET	123	721#												
SHONOD	124	726#												
SHOSPE	125	703#												
SHOTAB	120#	678												
SHOTRA	126	712#												
SLOG	686	695#												
SNET	79#	689	722											
SNOD	79#	735												
SOUT%	481	867#												
SOUT%1	871#	874												
SPEAR	78#	587	659	704										
SSPE	687	704#												
STRA	688	713#												
T%TEXT	280	291	368	409	424	487	573	599	671	698	707	716	732	741
T1	797	802	851	852	855	896	904	908	910	937	939	949	964	967
T2	244	245	250	293	463	498	499	516	517	541	543	548	549	550
T3	625	767	929	931	958	959	961							
T4	494	501	513	520	522	525								
TAKABT	441	474	486	540										
TAKE1	437	441	465	474	480	539	549	860	867	890	916			
TAKE3	803	807#	950	968										
TAKE6	748	778#												
TAKE01	774	797#												
TAKEOF	773	792	802#											
TAKFD	820	828#												
TAKFOB	819#	944												
TAKIFN	159	163#	765	786	787									
TOPLVL	158#	790												
TRACE	216#	342	740	766	770	793	807	814	821	828	938	946	965	
TXT.FN	286	295	336#	368	599	671	732	741	808	815	966			
TXTARG	78#	593	665	713										
VEDIT	894													
\$RETF	78#	248												
\$RETIF	37													
	65	822	825	830										
	65	567												

SEQ 0081

\$RETIT	65	895												
\$RETT	65	569	826	831										
%MOD	133	894												
%FB.BS	160	172	183	195										
%FD.LE	164	176	187	199										
%FWMAS	133	134	143	144	145	146	147	148	149	159	171	182	188	189
	190	194	200	201	202									
%IB.NP	136													
%IT.OC	135													
%RHMAS	142													
..0001	90#	100												
..0002	90	100#												
..0003	104#	108												
..0004	104	108#												
..0005	112#	116												
..0006	112	116#												
..0007	120#	127												
..0010	120	127#												
..0011	861	861#												
..0012	868#													
..0013	891	891#												
..0014	917	917#												
..BLOC	132#	137	138	141#	150	151	158#	161	162	163#	165	166	170#	173
	174	175#	177	178	181#	184	185	186#	191	192	193#	196	197	198#
	203	204												
..BSIZ	132	132#	133	134	135	136	137	138	141	141#	142	143	144	145
	146	147	148	149	150	151	158	158#	159	160	161	162	163	163#
	164	165	166	170	170#	171	172	173	174	175	175#	176	177	178
	181	181#	182	183	184	185	186	186#	187	188	189	190	191	192
	193	193#	194	195	196	197	198	198#	199	200	201	202	203	204
..MX1	389#	389	390	610#	610	611	625#	625	626	752#	752	753	771#	771
	772													
..MX2	389#	389	390	610#	610	611	625#	625	626	752#	752	753	771#	771
	772													
..T	138#	138	151#	151	162#	162	166#	166	174#	174	178#	178	185#	185
	192#	192	197#	197	204#	204								
..T0	132#	134	134#	138	141#	142	142#	151	158#	159	159#	162	163#	164
	164#	166	170#	171	171#	174	175#	176	176#	178	181#	182	182#	185
	186#	187	187#	192	193#	194	194#	197	198#	199	199#	204		
..T1	132#	135	135#	136	136#	138	141#	151	158#	160	160#	162	163#	166
	170#	172	172#	174	175#	178	181#	183	183#	185	186#	188	188#	192
	193#	195	195#	197	198#	200	200#	204						
..T10	141#	147	147#	151	163#	166	175#	178						
..T11	141#	149	149#	151	163#	166	175#	178						
..T12	163#	166	175#	178										
..T13	163#	166	175#	178										
..T14	163#	166	175#	178										
..T15	163#	166	175#	178										
..T16	163#	166	175#	178										
..T17	163#	166	175#	178										
..T2	132#	138	141#	148	148#	151	158#	162	163#	166	170#	174	175#	178
	181#	185	186#	189	189#	192	193#	197	198#	201	201#	204		
..T20	163#	166	175#	178										

SEQ 0082

..T21	163#	166	175#	178										
..T22	163#	166	175#	178										
..T23	163#	166	175#	178										
..T24	163#	166	175#	178										
..T25	163#	166	175#	178										
..T26	163#	166	175#	178										
..T27	163#	166	175#	178										
..T3	132#	138	141#	143	143#	151	158#	162	163#	166	170#	174	175#	178
	181#	185	186#	190	190#	192	193#	197	198#	202	202#	204		
..T30	163#	166	175#	178										
..T31	163#	166	175#	178										
..T32	163#	166	175#	178										
..T33	163#	166	175#	178										
..T34	163#	166	175#	178										
..T35	163#	166	175#	178										
..T36	163#	166	175#	178										
..T37	163#	166	175#	178										
..T4	132#	138	141#	144	144#	151	158#	162	163#	166	170#	174	175#	178
	181#	185	193#	197	198#	204								
..T40	163#	166	175#	178										
..T41	163#	166	175#	178										
..T42	163#	166	175#	178										
..T43	163#	166	175#	178										
..T44	163#	166	175#	178										
..T45	163#	166	175#	178										
..T46	163#	166	175#	178										
..T47	163#	166	175#	178										
..T5	132#	133	133#	138	141#	145	145#	151	163#	166	175#	178		
..T50	163#	166	175#	178										
..T51	163#	166	175#	178										
..T52	163#	166	175#	178										
..T53	163#	166	175#	178										
..T54	163#	166	175#	178										
..T55	163#	166	175#	178										
..T56	163#	166	175#	178										
..T57	163#	166	175#	178										
..T6	141#	151	163#	166	175#	178								
..T60	163#	166	175#	178										
..T61	163#	166	175#	178										
..T62	163#	166	175#	178										
..T63	163#	166	175#	178										
..T64	163#	166	175#	178										
..T65	163#	166	175#	178										
..T66	163#	166	175#	178										
..T67	163#	166	175#	178										
..T7	141#	146	146#	151	163#	166	175#	178						
..T70	163#	166	175#	178										
..T71	163#	166	175#	178										
..T72	163#	166	175#	178										
..T73	163#	166	175#	178										
..TMSK	133#	133	134#	134	135#	135	136#	136	142#	142	143#	143	144#	144
	145#	145	146#	146	147#	147	148#	148	149#	149	159#	159	160#	160
	164#	164	171#	171	172#	172	176#	176	182#	182	183#	183	187#	187

..TSA1	188#	188	189#	189	190#	190	194#	194	195#	195	199#	199	200#	200
..TVAL	201#	201	202#	202	890#	890	916#	916						
	860#	860	867#	867	135#	135	136#	136	142#	142	143#	143	144#	144
	133#	133	134#	134	147#	147	148#	148	149#	149	159#	159	160#	160
	145#	145	146#	146	172#	172	176#	176	182#	182	183#	183	187#	187
	164#	164	171#	171	190#	190	194#	194	195#	195	199#	199	200#	200
	188#	188	189#	189										
	201#	201	202#	202										
..TX1	239#	239	240	824#	824	825	927#	927	928					
..TX2	239#	240	824#	824										
..TXB	894#	894												
..TXC	894#	894												
..TXEF	894#	894												
..TXEG	894#	894												
..TXF	894#	894												
..TXP	894#	894												
..XX	85#	85	348#	348	353#	353	362#	362	364#	364	530#	530	533#	533
	579#	579	600#	600	620#	620	641#	641	651#	651	678#	678	726#	726
	728#	728	745#	745	762#	762	784#	784	836#	836				
.A13	860	868	890	916										
.A14	860	868	890	916										
.A15	860	868	890	916										
.A16	860	867	868	890	916									
.ACB	860#	860	861#	861	867#	867	890#	890	891#	891	916#	916	917#	917
.ACM	860#	860	861	867#	867	868	890#	890	891	916#	916	917		
.APRID	315													
.CHCRT	509	519	959											
.CHLFD	511	521												
.CMABC	147													
.CMABP	146													
.CMBFP	143													
.CMCFM	85													
.CMCNT	145													
.CMFIL	620	762												
.CMFLG	142													
.CMGJB	141	149	220	338										
.CMIFI	784													
.CMINI	348													
.CMIOJ	247	344	612	754										
.CMKEY	353	530	579	651	678									
.CMNOI	362	600	726	745	836									
.CMNUM	364	728												
.CMOFI	641													
.CMPTR	144	957												
.CMRTY	148	346												
.CMTOK	533													
.CONNE	91	362#												
.CPUTY	299													
.DDT	92	376#												
.DISAB	93	651#												
.ENABL	94	579#												
.FDEXT	202	638	781											
.FDLEN	164	176	187	199										

SEQ 0084

.FDNAM	201	636	779											
.FDPPN	198	199												
.FDSTG	186	187	188	189	190	561	623	765						
.FDSTR	200	560	640	783										
.FHSLF	288	292	393	396										
.GJDEV	619	761												
.GJEXT	617	759												
.GJGEN	611	753												
.GJNAM	615	757												
.GJRTY	223													
.GJSRC	613	755												
.HELP	96	430#												
.JBDDT	380													
.JSAOF	625													
.NVR	860#	860	861	861#	867#	867	868	890#	890	891	891#	916#	916	917
	917#													
.POPJ	65	872												
.PRIIN	240													
.PRIOU	240													
.PSECT	239	240	389	390	610	611	625	626	752	753	771	772	824	825
	860	867	890	916	927	928								
.QUIT	95	97	836#											
.RDIOJ	248													
.RETF	65													
.RETT	65													
.SAVE2	867													
.SHOW	98	678#												
.STOP	893													
.TAKE	99	740#												
.TQCHT	894													

SEQ 0085

BLDO.	132	138	141	151	158	162	163	166	170	174	175	178	181	185
	186	192	193	197	198	204								
CLOSF%	404	810												
CMDERR	368	599	671	732	741									
EPCAP%	294													
ERJMP	392	395	405	627	630	769	773	811						
FLD	85	133	134	135	136	142	143	144	145	146	147	148	149	159
	160	164	171	172	176	182	183	187	188	189	190	194	195	199
	200	201	202	348	353	362	364	530	533	579	600	620	625	641
	651	678	726	728	745	762	771	784	836	894				
FLDDB.	85	348	353	362	364	530	533	579	600	620	641	651	678	726
	728	745	762	784	836									
GET%	394													
GETAB%	316													
GETTAB	237	310												
GLOB	65													
GTJFN%	391													
GTSTS%	823													
GUIDE	362	726												
IONEOU	883													
ITEXT	323	696	697	894										
JFNS%	626	768												
JUMPF	65	453	496	792	848	925	945							
JUMPT	65	447	450	504										
KEYTAB	91	92	93	94	95	96	97	98	99	105	106	107	113	114
	115	121	122	123	124	125	126							
LSTOF.	65	137	150	161	165	173	177	184	191	196	203	281	292	325
	368	410	425	488	574	599	671	696	697	699	708	717	732	741
	798	803	852	853	856	861	891	894	897	905	909	911	917	938
	940	950	965	968										
LSTON.	138	151	162	166	174	178	185	192	197	204	861	868	891	894
	917													
MERGE.	416													
MOVX	389	610	624	752	771									
NAME	37													
OPENF%	772													
PBOUT%	885													
PJRST	65	489	673											
POS	85	133	134	135	136	142	143	144	145	146	147	148	149	159
	160	164	171	172	176	182	183	187	188	189	190	194	195	199
	200	201	202	348	353	362	364	530	533	579	600	620	625	641
	651	678	726	728	745	762	771	784	836	894				
PROLOG	64													
RLJFN%	629													
RPCAP%	289													
SET0.	133	134	135	136	142	143	144	145	146	147	148	149	159	160
	164	171	172	176	182	183	187	188	189	190	194	195	199	200
	201	202												
SEVEC%	398													
SKIPF	909													
SKIPT	907	953												
STORE	636	638	640	779	781	783								
TXNE	239	927												

SEQ 0086

TXNN	824													
\$BUILD	132	141	158	163	170	175	181	186	193	198				
\$CALL	256	272	277	280	291	299	349	354	356	363	365	368	370	376
	409	424	431	432	446	449	452	459	470	481	487	495	503	566
	573	580	586	592	599	601	621	622	642	645	652	658	664	671
	672	679	685	686	687	688	689	694	698	703	707	712	716	721
	727	729	732	734	741	746	763	764	785	788	791	797	802	813
	837	838	839	847	851	852	855	867	873	880	893	894	896	904
	905	908	910	924	937	939	944	949	952	955	962	964	967	
\$EOB	137	150	161	165	173	177	184	191	196	203				
\$ETAB	91#	100	105#	108	113#	116	121#	127						
\$RET	65#	65	313	319	410	526	542	574	588	594	660	666	690	699
	708	717	798	853	856	884	886	902	911	932	940			
\$RETF	65#													
\$RETIF	65#													
\$RETI	65#													
\$RETT	65#													
\$SAVE	860	867	890	916										
\$SET	133	134	135	136	142	143	144	145	146	147	148	149	159	160
	164	171	172	176	182	183	187	188	189	190	194	195	199	200
	201	202												
\$STAB	90	104	112	120										
\$STOP	893													
\$TEXT	278	291	368	409	424	487	573	599	671	698	707	716	732	741
	797	802	850	852	855	896	903	908	910	937	939	948	964	967
..CNS	894#	894												
..CNS2	894#	894												
..GCNS	894													
..ICNS	894													
..POP	861	891	917											
..PUSH	861	891	917											
..SET	133	134	135	136	142	143	144	145	146	147	148	149	159	160
	164	171	172	176	182	183	187	188	189	190	194	195	199	200
	201	202												
..STRO	133	134	135	136	142	143	144	145	146	147	148	149	159	160
	164	171	172	176	182	183	187	188	189	190	194	195	199	200
	201	202												
..TEXT	894													
..TSAC	860	867	890	916										
..TX	239	824	927											
..TXCO	894													
..TXC1	894													
..TXQ	894													
..TXQ0	894													
.NV1	861#	861	891#	891	917#	917								
.NV2	891#	891	917#	917										

SEQ 0087

```

1      ;Z:<GSCOTT.DFCID>DFCIDM.MAC.300 6-Sep-85 09:48:19, Edit by GSCOTT
2      ;Length of SDCBLK must be big enough to handle TTIBUF, but not too big because
3      ;it crashes the HSC50.
4      ;Z:<GSCOTT.DFCID>DFCIDM.MAC.290 4-Sep-85 18:37:38, Edit by GSCOTT
5      ;Correct spacing on SHOW NETWORK command.
6      ;Z:<GSCOTT.DFCID>DFCIDM.MAC.289 4-Sep-85 18:35:52, Edit by GSCOTT
7      ;Copy terminal input to the log file.
8      ;Z:<GSCOTT.DFCID>DFCIDM.MAC.287 4-Sep-85 17:36:04, Edit by GSCOTT
9      ;Report to spear, default spear reporting flag to -1
10     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.285 4-Sep-85 14:45:48, Edit by GSCOTT
11     ;Don't overrun code with SDCBLK data.
12     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.284 4-Sep-85 14:42:42, Edit by GSCOTT
13     ;^C interrupts don't work on the -20 when you also interrupt for all chars.
14     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.279 3-Sep-85 17:46:12, Edit by GSCOTT
15     ;Add ^C interrupt for TOPS-20 to disconnect.
16     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.277 3-Sep-85 17:36:50, Edit by GSCOTT
17     ;Remove WAITF, it isn't needed any more
18     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.273 3-Sep-85 17:08:16, Edit by GSCOTT
19     ;Use mnemonics for the DUP opcodes.
20     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.272 3-Sep-85 16:55:04, Edit by GSCOTT
21     ;Set the control-C trap on the -10 to disconnect, at priority level 3
22     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.271 3-Sep-85 16:13:17, Edit by GSCOTT
23     ;GLXKBD wants me to use 7 bit bytes in the KXTXTI routine USTOC at USTO.1,
24     ;since 8 bit byte sizes cause it to loop (it tries to backup a byte pointer
25     ;by doing a SOS then 4 IBPs!).
26     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.268 3-Sep-85 11:38:46, Edit by GSCOTT
27     ;Need to send response messages as the SDI messages come in
28     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.265 30-Aug-85 21:26:05, Edit by GSCOTT
29     ;Need to send responses back to HSC: bug at MSRV2
30     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.256 30-Aug-85 18:14:01, Edit by GSCOTT
31     ;Move interrupt code up to the important stuff area.
32     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.254 30-Aug-85 18:07:40, Edit by GSCOTT
33     ;Don't send a linefeed to the HSC (bug in TOPS-20 mode only)
34     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.249 30-Aug-85 14:45:23, Edit by GSCOTT
35     ;Need to break out of wait loop after message opcode 7 recieved.
36     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.242 30-Aug-85 13:52:03, Edit by GSCOTT
37     ;Echo a CRLF in response to the ^Y being typed.
38     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.235 30-Aug-85 13:38:28, Edit by GSCOTT
39     ;Remove flags: CBROK, CLIS, OKSEND, ONLIN, NODDIS, PBROK, CAVAIL, LCRED
40     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.230 30-Aug-85 13:21:28, Edit by GSCOTT
41     ;Get around a bug in GLXLIB-10: output LF if broke on CR.
42     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.226 30-Aug-85 12:32:45, Edit by GSCOTT
43     ;Give a little message when disconnect complete.
44     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.225 30-Aug-85 12:29:54, Edit by GSCOTT
45     ;Remove HIDE flag, never used really, and clean up message service int han.
46     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.212 30-Aug-85 11:52:22, Edit by GSCOTT
47     ;Set TOPS-10 interrupt levels (I hope they work)
48     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.209 30-Aug-85 11:51:36, Edit by GSCOTT
49     ;Remove use of ETFLG, instead call CKDUST when you want to check the status.
50     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.207 30-Aug-85 11:50:08, Edit by GSCOTT
51     ;Change interrupt levels around so that message can be send from ter/tim ints
52     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.179 30-Aug-85 09:49:55, Edit by GSCOTT
53     ;More changes to terminal interrupt handler to loop on ITY interrupts properly.
54     ;Z:<GSCOTT.DFCID>DFCIDM.MAC.165 29-Aug-85 22:07:07, Edit by GSCOTT
55     ;Add removal of TOPS-10 interrupt stuff at RESTOR.

```

```

56 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.161 29-Aug-85 21:50:52, Edit by GSCOTT
57 ;Set I/O JFNs from DFCIDP for KXTXTI.
58 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.160 29-Aug-85 21:45:47, Edit by GSCOTT
59 ;Don't do a JSYS on the 10 at INIT1.
60 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.157 29-Aug-85 21:33:03, Edit by GSCOTT
61 ;Try KXTXTI in GLXLIB to simulate TEXTI JSYS on the 10 and 20.
62 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.139 29-Aug-85 19:35:04, Edit by GSCOTT
63 ;Start implementation of TOPS-10 code
64 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.138 29-Aug-85 19:34:38, Edit by GSCOTT
65 ;Don't do anything fancy at disconnect time- its a waste.
66 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.130 29-Aug-85 17:41:23, Edit by GSCOTT
67 ;Improve FLAGIT to asciize the function code that failed.
68 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.127 29-Aug-85 17:26:30, Edit by GSCOTT
69 ;If we get invalid CID when the connection is being closed, it is closed.
70 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.125 29-Aug-85 17:13:17, Edit by GSCOTT
71 ;Remove DEBUG mode - it doesn't work and its not worth fixing.
72 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.119 29-Aug-85 15:44:04, Edit by GSCOTT
73 ;Remove extra .LBxxx references and words.
74 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.118 29-Aug-85 15:41:13, Edit by GSCOTT
75 ;Remove unused SSCRM routine, TIOBUF, OBUFCT storage
76 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.111 29-Aug-85 14:07:08, Edit by GSCOTT
77 ;Intern CHNTAB, LEVTAB, PIVECT, use GLXLIB to turn on/off interrupts
78 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.104 28-Aug-85 08:23:10, Edit by GSCOTT
79 ;Move TTX to immediately after SIBE.
80 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.102 27-Aug-85 20:26:02, Edit by GSCOTT
81 ;Remove EXSPGM since it isn't used here or supported on HSC50.
82 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.100 27-Aug-85 17:07:02, Edit by GSCOTT
83 ;Get rid of SOBF at DISP2- not needed
84 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.90 27-Aug-85 11:19:26, Edit by GSCOTT
85 ;Add TOPS-10 SCS UUD error codes
86 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.87 26-Aug-85 22:06:53, Edit by GSCOTT
87 ;Remove YESNO
88 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.86 26-Aug-85 22:02:55, Edit by GSCOTT
89 ;Remove PORNUM
90 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.81 26-Aug-85 21:44:37, Edit by GSCOTT
91 ;Fix PRRCD1 and PRRCD2.
92 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.64 26-Aug-85 19:10:36, Edit by GSCOTT
93 ;Use mnemonic for channels generated from ASSCHN macro.
94 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.62 26-Aug-85 18:59:52, Edit by GSCOTT
95 ;DEBRKing with stuff on stack from $SAVE causes pdl overflows.
96 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.44 26-Aug-85 15:49:47, Edit by GSCOTT
97 ;Fix RUTIME to not save S1 and to get uptime on the -10 right.
98 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.40 23-Aug-85 23:28:37, Edit by GSCOTT
99 ;Make $SCS put error code in SCSERC, fix up FLAGIT
100 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.28 23-Aug-85 21:49:15, Edit by GSCOTT
101 ;Fix AOS -n(P), clean up $TEXTs, etc.
102 ;Z:<GSCOTT.DFCID>DFCIDM.MAC.15 23-Aug-85 16:51:11, Edit by GSCOTT
103 ;Start GLXLIB conversion

```

```

SEARCH DFCIDT
NAME (DFCIDM,\DECVER,\VEDIT,DUP Driver Main Module)^

```

COMMENT \

COPYRIGHT (C) 1985

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL EQUIPMENT CORPORATION.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE IN EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL EQUIPMENT CORPORATION.

DFCIDM DUP Driver Main Module version 1(12) MACRO %53B(1242) 12:08 6-Sep-85 Page 2
DFCIDM MAC 6-Sep-85 10:19 Assembler Directives

SEQ 0091

```
130          SUBTTL  Assembler Directives
131
132          NOSYM
133          SALL
134          .DIREC  FLBLST
135
136          SEARCH  MONSYM,MACSYM,UUOSYM,GLXMAC
137
138          PROLOG  (DFCIDM)
139
140          ;Internal symbols
141
142          INTERN  SNET,SNOD,CONN
143          INTERN  NODNUM,TRACE,SPEAR,CHNTAB,LEVTAB,PIVECT,XTARG
144
145          ;External symbols
146
147          EXTERN  MONTYP,LOGIFN
148          EXTERN  BOUT%%,LOUT%%,PCRLF
```

```

149          SUBTTL Connect Command
150
151          ;* CONN - Entered by the "CONNECT" command from the PARSE
152
153          ;Enable system interrupts and SCA interrupt channels
154
155          000000' 260 17 0 00 001547' CONN: $CALL SYSINT ;Enable system interrupt channels
156          000001' 260 17 0 00 002263'      $CALL SCAINT ;Enable SCA interr for same channels
157          000002' 254 00 0 00 000050'      JRST CONX ; Error, return to PARSE
158
159          ;Connect to desired node. Synchronize to DUP Server to allow one
160          ;process to communicate with another process.
161
162          000003' 260 17 0 00 000700'      $CALL CONNEC ;Connect to selected node
163          000004' 254 00 0 00 000050'      JRST CONX ; Error, return to PARSE
164
165          ;Send GET DUST STATUS DUP command to establish DUP characteristics
166
167          000005' 260 17 0 00 001212'      $CALL DUSTAT ;Send DUP - GET DUST STATUS
168          000006' 254 00 0 00 000043'      JRST LOOPX ; Error, disconnect
169          000007' 260 17 0 00 001254'      $CALL CKDUST ;No, evaluate returned DUST Status
170          000010' 336 00 0 00 005667'      SKIPN LODMED ;Does HSC have load media?
171          000011' 254 00 0 00 000043'      JRST LOOPX ; no, disconnect
172          000012' 336 00 0 00 005672'      SKIPN SNDDIM ;Yes, does DUP support SEND DATA IMMED?
173          000013' 254 00 0 00 000043'      JRST LOOPX ; no, disconnect
174
175          ;Send the EXECUTE LOCAL PROGRAM DUP command to run PSEUDO which is the local
176          ;terminal emulator for the HSC50 (virtual LA12). Then send ^C to get the HSC50
177          ;prompt.
178
179          CON5: $TEXT (,<
180          000014' 260 17 0 00 000000* To Disconnect and return to command mode, type a single Control-D>)
181
182          000016' 260 17 0 00 000731'      $CALL EMULAT ;Run the local program PSEUDO
183          000017' 254 00 0 00 000043'      JRST LOOPX ; error, disconnect
184
185          000020' 205 01 0 00 014000      MOVSI S1,(<BYTE (7).TICCC,0>) ;Load a ^C, null
186          000021' 202 01 0 00 005715'      MOVEM S1,TTIBUF ; and store in TTIBUF
187          000022' 201 01 0 00 000001'      MOVEI S1,1 ;Load character count of one
188          000023' 202 01 0 00 005714'      MOVEM S1,IBUFCT ; and store it there
189
190          000024' 260 17 0 00 001041'      $CALL SNDHSC ;Send ^C to HSC
191          000025' 254 00 0 00 000043'      JRST LOOPX ; error, disconnect
192
193          000026' 260 17 0 00 001520' CON7: $CALL INIT1 ;Init buffers, flags, timer, term

```

```

194
195      ; Loop and wait for characters from TTY input
196      ;
197      ; Use the WAIT JSYS/HIBER UUD to sleep until interrupts occur.
198      ; Interrupt level restarts at LOOP1 when we want to disconnect.
199
200 000027' 202 17 0 00 005725' LOOP:  MOVEM  P,SAVEP      ;Save the current stack pointer please
201 000030' 332 00 0 00 000000*   SKIPE  MONTYP    ;Check monitor type
202 000031' 254 00 0 00 000036'   JRST   LOOP2      ;TOPS-20
203 000032' 400 01 0 00 000000    SETZ   S1,         ;Clear S1
204 000033' 047 01 0 00 000072    HIBER  S1,         ;Sleep forever
205 000034' 255 00 0 00 000000    JFCL                ;Ignore error
206 000035' 254 00 0 00 000037'   JRST   LOOP1      ;Woke up? Check things then sleep
207
208 000036' 104 00 0 00 000306   LOOP2: WAIT%          ;Wait forever on the 20
209
210 000037' 200 17 0 00 005725'   LOOP1: MOVE   P,SAVEP    ;Reload the stack pointer
211 000040' 332 00 0 00 005713'   SKIPE  DISFLG    ;Did user enter ^D?
212 000041' 254 00 0 00 000043'   JRST   LOOPX      ; Yes, restore all and return
213 000042' 254 00 0 00 000027'   JRST   LOOP      ;Repeat all
214
215      ;Here if we got disconnect character (or error)
216
217 000043' 336 00 0 00 005671'   LOOPX: SKIPN  ACTIVE    ;DUP Server ACTIVE?
218 000044' 254 00 0 00 000047'   JRST   CON20      ; no, just disconnect and leave
219 000045' 260 17 0 00 001126'   $CALL  ABORTT    ;Yes, do ABORT to go to IDLE state
220 000046' 255 00 0 00 000000    NOP                    ; ignore err, disconn and ret to PARSER
221 000047' 260 17 0 00 001167'   CON20: $CALL  DISCON    ;Disconnect and close the connection
222 000050' 260 17 0 00 001613'   CONX:  $CALL  RESTOR    ;Restore all back to normal
223      $TEXT      (<
224 000051' 261 17 0 00 000014*   Disconnected from node ^D/NODNUM/.)> ;Output CRLF
225 000053' 263 17 0 00 000000    $RET                ; and return to PARSER

```

```

226          SUBTTL Interrupt Channel/Level/Vector Tables
227
228          ;TOPS-20 interrupt tables
229
230          000000          ONCHNS==0          ;Channels enabled: none
231          000000          CHNCNT==0          ;Start with channel 0
232
233          DEFINE ASSCHN(LEVEL,ADDR,CNAME,LNAME),<
234          XWD          LEVEL,ADDR          ;;Set interrupt level,,address
235          CNAME==CHNCNT          ;;Set interrupt channel symbol
236          LNAME==LEVEL          ;;Set interrupt level symbol
237          ONCHNS==ONCHNS!1B<CHNCNT>          ;;Count this channel as one to activate
238          CHNCNT==CHNCNT+1          ;;Bump up channel count
239          IFE          CHNCNT-6,<
240          CHNCNT==^D23          ;;If first 6 used, get to last 12
241          BLOCK          ^D23-6          ;; and fill in the panic channels
242          >;end of IFE CHNCNT-6
243          >;End of DEFINE ASSCHN
244
245          CHNTAB: ASSCHN (1,TTINP,TTICHN,TTILEV) ;Terminal input character available
246          ASSCHN (2,DMA SRV,DMACHN,DMALEV) ;SCS% DMA finished interrupts
247          ASSCHN (2,EVT SRV,EVTCHN,EVTLEV) ;SCS% EVENT interrupts
248          ASSCHN (2,MSG SRV,MSGCHN,MSGLEV) ;SCS% MESSAGE interrupts
249          ASSCHN (2,DG SRV,DGMCHN,DGMLEV) ;SCS% DATAGRAM interrupts
250          ASSCHN (3,ETIME,TIMCHN,TIMLEV) ;TIMER% interrupts
251          ASSCHN (1,CCIN20,CCCCHN,CCCLEV) ;Control-C traps
252          BLOCK ^D36-CHNCNT ;Zero rest of table
253
254          LEVTAB: 0,,PC1 ;Address to save PC for Lv 1
255          0,,PC2 ; ditto for Lv 2
256          0,,PC3 ; ditto for Lv 3
257
258          PC1: 0 ;Ret addr for Lv 1 interrupt (highest)
259          PC2: 0 ; ditto for Lv 2
260          PC3: 0 ; ditto for Lv 3
  
```

```

261
262      ;TOPS-10 Interrupt Vector
263
264      PIVECT: SCOFFS==.-PIVECT      ;Offset for SCS interrupts in table
265      000126' 000000 000372'      EXP SCSI10      ;..PSVNP SCS interrupt routine
266      000127' 000000 000000      EXP 0      ;..PSVOP Old PC
267      000130' 000000 777743      XWD 0,.,PCSCS      ;..PSVFL SCS interrupts
268      000131' 000000 000000      EXP 0      ;..PSVIS interrupt status
269      000132' 000000 000004'      TIOFFS==.-PIVECT      ;Offset for TTY input available
270      000133' 000000 000150'      EXP TTYI10      ;..PSVNP TTY interrupt routine
271      000134' 000000 000000      EXP 0      ;..PSVOP Old PC
272      000135' 000000 200010      XWD 0,PS.RID!PS.RIA      ;..PSVFL input interrupts
273      000136' 000000 000000      EXP 0      ;..PSVIS interrupt status
274      000137' 000000 000010      TIOFFS==.-PIVECT      ;Offset for PITMR interrupts
275      000138' 000000 000324'      EXP TIMI10      ;..PSVNP Timer interrupt routine
276      000139' 000000 000000      EXP 0      ;..PSVOP Old PC
277      000140' 000000 777776      XWD 0,.,PCTMR      ;..PSVFL timer interrupts
278      000141' 000000 000000      EXP 0      ;..PSVIS interrupt status
279      000142' 000000 000014'      CCOFFS==.-PIVECT      ;Offset for control C trap
280      000143' 000000 000300'      EXP CCIN10      ;..PSVNP New PC
281      000144' 000000 000000      EXP 0      ;..PSVOP Old PC
282      000145' 000000 777775      XWD 0,.,PCSTP      ;..PSVFL ^C interrupts
283      000146' 000000 000000      EXP 0      ;..PSVIS interrupt status

```

```

284          SUBTTL Terminal Interrupt Handlers
285
286          ;* TTYINP - TOPS-20 Terminal input interrupt handler
287
288          000146' 260 17 0 00 000152'  TTYINP: $CALL TT00          ;Do the work, use $CALL for $SAVEing
289          000147' 104 00 0 00 000136'  DEBRK%          ;Dismiss interrupt
290
291          ;* TTYI10 - TOPS-10 Terminal input interrupt handler
292
293          000150' 260 17 0 00 000152'  TTYI10: $CALL TT00          ;Do the work
294          000151' 047 00 0 00 000137'  DEBRK.          ;Return from interrupt
295
296          ;* TT00 - common routine to get input from terminal
297
298          000152' 304 00 0 00 000000'  TT00:  $SAVE <S1,S2>          ;Save ACs
299          000157' 332 00 0 00 005705'  $KIPE TRACE          ;Skip if not trace
300          000160' 260 17 0 00 000051*  $TEXT (<,TT00 - Terminal input available interrupt>)
301
302          ;Check if input buffer empty. If not, get some characters.
303
304          000162' 260 17 0 00 000253'  TT10:  $CALL TTCHRC          ;See if input buffer empty
305          000163' 326 00 0 00 000172'  JUMPT TT1          ;Jump if something there
306
307          ;Finished with this interrupt. See if we need to disconnect then return.
308
309          000164' 336 00 0 00 005713'  SKIPN DISFLG          ;Disconnect flag set?
310          000165' 263 17 0 00 000000'  $RET          ;No
311          000166' 200 01 0 00 006015'  MOVE S1,[1B5+LOOP1]          ;User mode bit+new PC
312          000167' 202 01 1 00 000122'  MOVEM S1,@LEVTAB+TTILEV-1          ;Yes, set the return PC
313          000170' 202 01 0 00 000133'  MOVEM S1,PIVECT+TTOFFS+.PSVOP ; to wake up lower loop
314          000171' 263 17 0 00 000000'  $RET
  
```

```

315
316 ;Here to read characters from terminal using GLXLIB's TEXTI emulator.
317
318 000172' 200 01 0 00 006016' TT1: MOVE S1,[POINT 7,TTIBUF] ;Byte ptr to where to put input
319 000173' 202 01 0 00 000241' MOVEM S1,XTARG+.RDBBP ; to arg block
320 000174' 202 01 0 00 000243' MOVEM S1,XTARG+.RDBFP
321 000175' 202 01 0 00 000246' MOVEM S1,XTARG+.RDBKL
322 000176' 201 01 0 00 000024' MOVEI S1,TTISIZ ;# bytes avail in string
323 000177' 202 01 0 00 000242' MOVEM S1,XTARG+.RDBBC ; to arg block
324 000200' 201 01 0 00 000236' MOVEI S1,XTARG ;Addr of arg block
325 000201' 260 17 0 00 000000* $CALL K%XTI ;Perform/simulate TEXTI JSYS
326 000202' 322 00 0 00 000233' JUMPF TT8 ;Error
327
328 ;Compute characters stored then see if a break character terminated input.
329
330 000203' 201 01 0 00 000024' MOVEI S1,TTISIZ ;Load maximum bytes
331 000204' 274 01 0 00 000242' SUB S1,XTARG+.RDBBC ;Less returned available bytes left
332 000205' 202 01 0 00 005714' MOVEM S1,IBUFCT ;Equals byte count
333
334 000206' 200 01 0 00 000237' MOVE S1,XTARG+.RDFLG ;Get returned flags
335 000207' 607 01 0 00 000040' TLNN S1,(RD%BTM) ;Did Break char terminate the input?
336 000210' 254 00 0 00 000226' JRST TT4 ;No, ran out of space, send it now
337
338 ;Break character typed. Check it to see if it was the disconnect character.
339
340 000211' 200 01 0 00 000241' MOVE S1,XTARG+.RDDEP ;Get updated byte ptr on ret
341 000212' 135 01 0 00 000001' LDB S1,S1 ;Get character
342 000213' 306 01 0 00 000012' CAIN S1,.CHLFD ;Is it a linefeed?
343 000214' 254 00 0 00 000162' JRST TT10 ;Yes, must be after a CR
344 000215' 306 01 0 00 000004' CAIN S1,'D'-100 ;Was the last character a ^D ?
345 000216' 254 00 0 00 000231' JRST TT3 ; no, must be ^Y or CR, set break flg
346 000217' 306 01 0 00 000031' CAIN S1,'Y'-100 ;Was it a control-Y?
347 000220' 260 17 0 00 000000* $CALL PCRLF ;Yes, echo a CRLF to terminal
348 000221' 302 01 0 00 000015' CAIE S1,.CHCRT ;Was it a return?
349 000222' 254 00 0 00 000226' JRST TT4 ;No, send it
350 000223' 201 01 0 00 000012' MOVEI S1,.CHLFD ;Load a linefeed
351 000224' 336 00 0 00 000030* SKIPN MONTYP ;-20?
352 000225' 260 17 0 00 000000* $CALL BOUT% ;no, a -10, get around GLXLIB-10 bug
353
354 ;Send the data to the HSC50.
355
356 000226' 260 17 0 00 001041' TT4: $CALL SNDHSC ;Turn off int, send string, turn on int
357 000227' 254 00 0 00 000231' JRST TT3 ;If an error, punt the connection
358 000230' 254 00 0 00 000162' JRST TT10 ;Look for more characters if needed
359
360 ;Here if a ^D typed
361
362 000231' 476 00 0 00 005713' TT3: SETOM DISFLG ;Yes, set disconnect flag
363 000232' 254 00 0 00 000162' JRST TT10 ;Loop

```

```

364
365 ;Here if an error was detected in GLXLIB's texti emulator.
366
367 000233' 260 17 0 00 000160* TT8: $TEXT (,<? ITYINP - error return from K%XTXI: ^E/S1/>)
368 000235' 254 00 0 00 000162' JRST IT10 ;Try again
369
370 ;TEXTI Argument Block
371
372 000236' 000000 000010 TXTARG: 10 ;# words of arg block to follow
373 000237' 004000 000000 RD%JFN ;Flag bits: following word = JFN
374 ; (Updated flags on return)
375 000240' 000100 000101 .PRIIN,,.PRIOU ;Input JFN,,output JFN
376 000241' 44 07 0 00 005715' POINT 7,TTIBUF ;Byte ptr to where input to be placed
377 ; (Updated byte ptr on return)
378 000242' 000000 000024 TTISIZ ;# bytes avail (updated on return)
379 000243' 44 07 0 00 005715' POINT 7,TTIBUF ;Byte ptr to begin of dest buffer
380 000244' 000000 000000 0
381 000245' 000000 000247' BRKBLK ;Addr of 4 word block of break chars
382 000246' 44 07 0 00 005715' POINT 7,TTIBUF
383
384 000247' 020220 002000 BRKBLK: 1B4+1B10+1B13+1B25 ;Mask to break on ^D,LF,CR,^Y resp
385 000250' 000000 000000 0
386 000251' 000000 000000 0
387 000252' 000000 000000 0
388
389 ;* TTCHRC - common routine to check if something in input buffer
390 ; returns FALSE if nothing in buffer
391 ; returns TRUE if there is
392
393 000253' 336 00 0 00 000224* TTCHRC: SKIPN MONTYP ;TOPS-10?
394 000254' 254 00 0 00 000261' JRST TTCHR1 ;Yes
395
396 000255' 201 01 0 00 000100 MOVEI S1,.PRIIN ;TTY source designation
397 000256' 104 00 0 00 000102 SIBEX ;TTY input buffer empty?
398 000257' 324740 000000* $RETT ;Nope, return true
399 000260' 324740 000000* $RETF ;yes, return false
400
401 000261' 051 13 0 00 000000 TTCHR1: SKPINC ;Skip if character can be input
402 000262' 324740 000260* $RETF ;No character, return false
403 000263' 324740 000257* $RETT ;Character, return true

```

```

404
405      ;* CCIN20 - Here on ^C interrupt for TOPS-20
406
407      000264' 260 17 0 00 000266'  CCIN20: $CALL  CCIN21      ;Do the work
408      000265' 104 00 0 00 000136'  DEBRK%      ;Return
409
410      000266' 304 00 0 00 000000'  CCIN21: $SAVE  <S1,S2>      ;Save a couple
411      000273' 201 01 0 00 000037'  MOVEI  S1,.,TICTI      ;Clear interrupts
412      000274' 104 00 0 00 000140'  DTI%      ;De-assign all type-in
413      000275' 200 03 0 00 006015'  MOVE  T1,[1B5+LOOP1]      ;User mode bit+wakeup PC
414      000276' 202 03 1 00 000120'  MOVEM T1,@LEVTAB+CCCLEV-1 ; to wake up lower loop
415      000277' 263 17 0 00 000000'  $RET
416
417      ;* CCIN10 - Here on ^C interrupt for TOPS-10
418
419      000300' 260 17 0 00 000302'  CCIN10: $CALL  CCIN11      ;Do the work
420      000301' 047 00 0 00 000137'  DEBRK.      ;Return
421
422      000302' 304 00 0 00 000000'  CCIN11: $SAVE  <S1,S2>      ;save an AC or two
423      000307' 332 00 0 00 000145'  SKIPE  PIVECT+CCOFFS+.PSVIS ;Was the ^C typed at a prompt?
424      000310' 051 03 0 00 006035'  OUTSTR [ASCII/^C/]      ;Yes, tell him a control C
425      000311' 051 11 0 00 000000'  CLRBFI      ;Clear input buffer
426      000312' 200 01 0 00 006041'  MOVE  S1,[PS.FRC![EXP <SIXBIT/TTY/>,<TTOFFS,.,PS.RID!PS.RIA>,<1,.,0>]]
427      000313' 047 01 0 00 000136'  PISYS. S1,      ;Disable terminal char avail ints
428      000314' 260 17 0 00 000233*  $TEXT  (, <? Can't remove TTY interrupts - error code ^O/S1/>)
429      000316' 476 00 0 00 005713'  SETOM  DISFLG      ;Disconnect now please
430      000317' 200 01 0 00 006015'  MOVE  S1,[1B5+LOOP1]      ;User mode bit+wakeup PC
431      000320' 202 01 0 00 000143'  MOVEM S1,PIVECT+CCOFFS+.PSVOP ; to wake up lower loop
432      000321' 263 17 0 00 000000'  $RET

```

```

433          SUBTTL Timeout Interrupt Handler
434
435          ;* ETIME - TOPS-20 Timer interrupt handler
436
437 000322' 260 17 0 00 000326' ETIME: $CALL   KEEPCH          ;Keep the connection open
438 000323' 104 00 0 00 000136      DEBRK%          ;Dismiss interrupt
439
440          ;Here for PITMR interrupts on the -10
441
442 000324' 260 17 0 00 000326' TIMI10: $CALL   KEEPCH          ;Keep the connection open
443 000325' 047 00 0 00 000137      DEBRK.          ;Dismiss interrupt
444
445          ;Here to keep connection open, called when TIMFLG nonzero
446
447 000326' 332 00 0 00 005705' KEEPCH: SKIPE   TRACE
448 000327' 260 17 0 00 000314'      $TEXT   (<,KEEPCH - Sending GET DUST STATUS to keep connection>)
449 000331' 260 17 0 00 001212'      $CALL   DUSTAT          ;Get DUST status
450 000332' 260 17 0 00 000327'      $TEXT   (<,<? KEEPCH - error in get DUST status>)
451
452          ;* INITIM - Initialize reservation timer
453
454          ; The timer is used to tick off 30 seconds. If a character is not rec'd by
455          ; that time, a GET DUST STATUS is issued so we dont loose the connection.
456
457 000334' 304 00 0 00 000000 INITIM: $SAVE   <S1,S2,T1>          ;Save some ac's
458 000342' 336 00 0 00 000253*      SKIPN   MONIYP          ;Skip if TOPS-20
459 000343' 254 00 0 00 000353*      JRST    INITI1          ;TOPS-10
460
461          MOVE   S1,[FHS LF,...TIMEL]          ;Process handle,,do elapsed time
462          MOVEI  S2,RESTIM          ;For 30 sec or so
463          MOVEI  T1,TIMCHN          ;Interrupt on channel 5
464          TIMER%          ;Invoke timer jsys
465          $TEXT   (<,<? INITIM - error in setting timer interrupt>)
466          $RET
467
468 000353' 200 01 0 00 006141' INITI1: MOVX   S1,PS.TMS+RESTIM          ;Load the time to sleep
469 000354' 047 01 0 00 000203      PITMR.  S1,          ;Cause a timed event in so many MS
470 000355' 260 17 0 00 000350*      $TEXT   (<,<? Can't set PITMR interrupt, error code ^O/S1/>)
471 000357' 263 17 0 00 000000      $RET          ;Return now please

```

```

472          SUBTTL  SCS Interrupt Handlers
473
474          ;* DMASRV - TOPS-20 SCS% DMA FINISHED interrupt service routine
475
476          000360' 260 17 0 00 000355*  DMASRV: $TEXT  (,<? Received unexpected DMA finished interrupt>)
477          000362' 104 00 0 00 000136  DEBRK%
478
479          ;* DGSRV - TOPS-20 SCS% DATAGRAM RECEIVED interrupt service routine
480
481          000363' 260 17 0 00 000360*  DGSRV:  $TEXT  (,<? Received unexpected datagram received interrupt>)
482          000365' 104 00 0 00 000136  DEBRK%
483
484          ;* EVTSRV - TOPS-20 SCS% EVENT interrupt service routine
485
486          000366' 260 17 0 00 000406'  EVTSRV: $CALL  EVTS0          ;Do the work
487          000367' 104 00 0 00 000136  DEBRK%          ;Dismiss interrupt
488
489          ;* MSGSRV - TOPS-20 SCS% message available interrupt service routine
490
491          000370' 260 17 0 00 000556'  MSGSRV: $CALL  MSGSR0        ;Do the work
492          000371' 104 00 0 00 000136  DEBRK%        ;Return from interrupt
493
494          ;* SCSI10 - TOPS-10 SCS interrupt handler.
495
496          000372' 260 17 0 00 000374'  SCSI10: $CALL  SCSI11        ;Do the work
497          000373' 047 00 0 00 000137  DEBRK.        ;Dismiss the interrupt
498
499          000374' 260 17 0 00 000000*  SCSI11: $SAVE  <P1>          ;Save an AC
500          000375' 336 07 0 00 000131'  SKIPN  P1,PIVECT+SCOFFS+.PSVIS ;Load interrupt status word
501          000376' 263 17 0 00 000000  $RET          ;Return if it was extra interrupt
502          000377' 603 07 0 00 040000  TXNE  P1,SC%EVA      ;Event pending?
503          000400' 254 00 0 00 000406'  JRST  EVTS0          ;Yes, dispatch to it
504          000401' 603 07 0 00 400000  TXNE  P1,SC%MVA      ;Message available?
505          000402' 254 00 0 00 000556'  JRST  MSGSR0        ;Yes, dispatch to that
506          000403' 260 17 0 00 000363*  $TEXT  (,<? SCSI10 - Unknown SCS interrupt cause, status ^0/P1/>)
507          000405' 263 17 0 00 000000  $RET          ;Return

```

```

508
509      ;* EVTS0 - Common SCS event interrupt handler
510
511      000406' 304 00 0 00 000000      EVTS0: $SAVE    <S1,S2>          ;Save ACs
512      000413' 332 00 0 00 005705'      $SKIPE    TRACE          ;Trace enabled?
513      000414' 260 17 0 00 000403*      $TEXT    (,<EVTSRV - Received Event Interrupt>)
514
515      000416' 260 17 0 00 002447'      EVTS1: $CALL    SSEVT          ;Get an entry off the event queue
516      000417' 254 00 0 00 000432'      JRST     EVTS4          ;Jsyst error, see if queue empty
517
518      000420' 200 01 0 00 002474'      MOVE     S1,EVTARG+.SQEVT      ;Get event code
519      000421' 301 01 0 00 000015'      CAIL     S1,.SEMAX          ;Defined?
520      000422' 201 01 0 00 000000'      MOVEI    S1,0              ;No force error mess
521      000423' 336 00 0 00 005705'      SKIPN    TRACE          ;Trace enabled?
522      000424' 254 00 0 00 000430'      JRST     EVTS3          ;no
523      MOVE     S2,[
524      [ASCIZ/Undefined code/] ;0
525      [ASCIZ/Virtual circuit broken/] ;1
526      [ASCIZ/Connect to listen/] ;2
527      [ASCIZ/Connection accepted/] ;3
528      [ASCIZ/Connection rejected/] ;4
529      [ASCIZ/Message send complete/] ;5
530      [ASCIZ/Little credit left/] ;6
531      [ASCIZ/Node went offline/] ;7
532      [ASCIZ/Node came online/] ;10
533      [ASCIZ/OK to send data/] ;11
534      [ASCIZ/Remote node initiated disconnect/] ;12
535      [ASCIZ/Port broke connection/] ;13
536      [ASCIZ/Undefined code/]](S1) ;14
537
538      000425' 200 02 0 01 006334'      $TEXT    (,<      Event (^O/EVTARG+.SQEVT/): ^T/(S2)/>)
539
540      000430' 260 17 1 01 000444'      EVTS3: $CALL    @EVNTBL(S1)      ;Set/Clr flags and get data from .SQDTA
541      000431' 254 00 0 00 000416'      JRST     EVTS1          ;Repeat until Queue empty
  
```

```

541
542 ;Here to check to see that event queue is empty and return if so.
543
544 000432' 200 02 0 00 005727' EVTS4: MOVE S2,SCSERC ;Load error code
545 000433' 302 02 0 00 602613 CAIE S2,SCSQIE ;TOPS-20 Event queue empty?
546 000434' 306 02 0 00 000025 CAIN S2,SSEQE% ;TOPS-10 Event queue empty?
547 000435' 304 00 0 00 000000 CAIA ;Yes
548 000436' 254 00 0 00 003014' JRST FLAGIT ;No, flag the error and return +1
549 000437' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
550 000440' 260 17 0 00 000426* $TEXT (<EVTSRV - Event Queue Empty>)
551 000442' 476 00 0 00 005656' SETOM EQEMPTY ;Set Q empty flag
552 000443' 263 17 0 00 000000 $RET ;Return
553
554 ;Table of routines to handle event interrupts
555
556 000444' 000000 000462' EVNTBL: IGNORE ;No such event (spurious int.)
557 000445' 000000 000463' VCBROK ;Virtual circuit broken
558 000446' 000000 000470' CONLIS ;Connect to listen
559 000447' 000000 000474' ACCEPT ;Connection accepted
560 000450' 000000 000502' REJECT ;rejected
561 000451' 000000 000534' TXDONE ;Msg/Dg send complete
562 000452' 000000 000541' LOCREL ;Little credit left
563 000453' 000000 000545' OFFLIN ;Node went offline
564 000454' 000000 000462' IGNORE ;Node came online
565 000455' 000000 000462' IGNORE ;Ok to send data
566 000456' 000000 000552' RDISCN ;Remote node initiated disconnect
567 000457' 000000 000554' PDISCN ;Port broke connection
568 000460' 000000 000462' IGNORE ;Credits available
569 000461' 000000 000462' IGNORE ;No such event
570
571 ;Event subroutines
572
573 000462' 263 17 0 00 000000 IGNORE: $RET ;Do nothing else about the event
574
575 000463' 402 00 0 00 005654' VCBROK: SETZM CONACC ;Virtual circuit broken
576 000464' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled
577 000465' 260 17 0 00 000440* $TEXT (< SBI = ^0/EVTARG+.SQDTA/>)
578 000467' 263 17 0 00 000000 $RET
579
580 000470' 332 00 0 00 005705' CONLIS: SKIPE TRACE ;Trace enabled
581 $TEXT (< Connect Data = ^0/EVTARG+.SQDTA/ ^0/EVTARG+.SQDTA+1/ ^0/EVTARG+.SQDT
582 000471' 260 17 0 00 000465* A+2/^0/EVTARG+.SQDTA+3/>)
583 000473' 263 17 0 00 000000 $RET
584
585 000474' 476 00 0 00 005654' ACCEPT: SETOM CONACC ;Connection accepted
586 000475' 336 00 0 00 005705' SKIPN TRACE ;Trace enabled
587 000476' 254 00 0 00 000501' JRST ACCX ;no
588 $TEXT (<Connect Data = ^0/EVTARG+.SQDTA/ ^0/EVTARG+.SQDTA+1/ ^0/EVTARG+.SQDTA+2/
589 000477' 260 17 0 00 000471* ^0/EVTARG+.SQDTA+3/>)
590 000501' 263 17 0 00 000000 ACCX: $RET

```

```

591
592 000502' 304 00 0 00 000000 REJECT: $SAVE <S1>
593 000506' 402 00 0 00 005654' SETZM CONACC ;Connection rejected
594 000507' 336 00 0 00 005705' SKIPN TRACE ;Trace enabled
595 000510' 254 00 0 00 000533' JRST REJX ;no
596 000511' 200 01 0 00 002475' MOVE S1,EVTARG+.SQDTA ;Get reason code
597 000512' 260 17 0 00 000477* $TEXT (,<Reason code = ^0/S1/>)
598 000514' 306 01 0 00 000001 CAIN S1,^D1
599 000515' 260 17 0 00 000512* $TEXT (,< = Match found for connect request>)
600 000517' 306 01 0 00 000012 CAIN S1,^D10
601 000520' 260 17 0 00 000515* $TEXT (,< = No match found for connect request>)
602 000522' 306 01 0 00 000022 CAIN S1,^D18
603 000523' 260 17 0 00 000520* $TEXT (,< = No resources to process connect request>)
604 000525' 306 01 0 00 000032 CAIN S1,^D26
605 000526' 260 17 0 00 000523* $TEXT (,< = Connection has been broken>)
606 000530' 306 01 0 00 000042 CAIN S1,^D34
607 000531' 260 17 0 00 000526* $TEXT (,< = Reserved>)
608 000533' 263 17 0 00 000000 REJX: $RET
609
610 000534' 476 00 0 00 005655' TXDONE: SETOM SBUF ;Msg/DG send complete
611 000535' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled
612 000536' 260 17 0 00 000531* $TEXT (,< Addr of sent buff = ^0/EVTARG+.SQDTA/>)
613 000540' 263 17 0 00 000000 $RET
614
615 000541' 332 00 0 00 005705' LOCRED: SKIPE TRACE ;Trace enabled
616 $TEXT (,< Number of credits req'd to restore calling programs credit threshold
617 = ^0/EVTARG+.SQDTA/>)
618 000544' 263 17 0 00 000000 $RET
619
620 000545' 402 00 0 00 005654' OFFLIN: SETZM CONACC ;Node went offline
621 000546' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled
622 000547' 260 17 0 00 000542* $TEXT (,< SBI of offline sys = ^0/EVTARG+.SQDTA/>)
623 000551' 263 17 0 00 000000 $RET
624
625 000552' 402 00 0 00 005654' RDISCN: SETZM CONACC ;Remote node init disconnect
626 000553' 263 17 0 00 000000 $RET
627
628 000554' 402 00 0 00 005654' RDISCN: SETZM CONACC
629 000555' 263 17 0 00 000000 $RET

```

```

630
631      ;* MSGSR0 - Common SCS message received interrupt service routine
632      ;
633      ; Evaluate the returned data in the SSRMG argument block.
634      ; Set BUFADR to the address of the returned message buffer.
635      ; Set BADNOD to -1 if returned node# not same as desired node#
636
637      000556' 304 00 0 00 000000 MSGSR0: $SAVE <S1,S2,T1,T2,T3,T4> ;Save the ACs
638      000567' 332 00 0 00 005705'      SKIPE TRACE ;Trace enabled?
639      000570' 260 17 0 00 000547*      $TEXT (<,MSGSRV - Received Message Available interrupt>)
640
641      000572' 260 17 0 00 002675' MSRV12: $CALL SSRMG ;Retrieve a message
642      000573' 254 00 0 00 000671'      JRST MSRV16 ; error, see if queue empty
643
644      ;Recieved message OK, remember buffer address, check returned node number.
645
646      000574' 200 01 0 00 002721'      MOVE S1,RMGARG+.SQARB ;Get addr of ret msg buffer
647      000575' 202 01 0 00 005703'      MOVEM S1,BUFADR ; and save it
648      000576' 402 00 0 00 005662'      SETZM BADNOD ;Not a bad node yet
649      000577' 550 01 0 00 002722'      HRRZ S1,RMGARG+.SQDFL ;Get node# only
650      000600' 316 01 0 00 005702'      CAMN S1,NODNUM ;Same as desired node?
651      000601' 254 00 0 00 000606'      JRST MSRV4 ; yes, continue
652      000602' 476 00 0 00 005662'      SETOM BADNOD
653      $TEXT (<,<? MSGSRV - Returned node in recieved message not same as desired node
654      000603' 260 17 0 00 000570*      Returned Node = ^D/S1/, Connected Node = ^D/NODNUM/.>)
655      000605' 254 00 0 00 000650'      JRST MSRV10 ;Re-queue buffer and exit
656
657      ;Check returned opcode to see if it is expected.
658
659      000606' 200 03 0 00 005703' MSRV4: MOVE T1,BUFADR ;Get command msg buffer address
660      000607' 132 02 0 00 006722'      LDB S2,[POINT 8,2(T1),7] ;Get opcode/endcode byte
661      000610' 306 02 0 00 000007'      CAIN S2,OPCSDI ;Is it SDI opcode from HSC?
662      000611' 254 00 0 00 000635'      JRST MSRV7 ; Yes, display data
663      000612' 200 01 0 00 005724'      MOVE S1,CUROPC ;No, get current opcode
664      000613' 271 01 0 00 000200'      ADDI S1,OPCRES ;Convert to the response opcode
665      000614' 316 02 0 00 000001'      CAMN S2,S1 ;Is it expected endcode?
666      000615' 254 00 0 00 000624'      JRST MSRV6 ; Yes, continue
667      000616' 306 02 0 00 000200'      CAIN S2,INVRES ;Invalid command?
668      000617' 260 17 0 00 000603*      $TEXT (<,<? MSGSRV - endcode = 200 invalid command issued>)
669      $TEXT (<,<? MSGSRV - endcode is not correct response
670      000621' 260 17 0 00 000617*      Received: ^O/S2/, Expected: ^O/S1/>)
671      000623' 254 00 0 00 000650'      JRST MSRV10 ;Re-queue buffer and exit
672
673      ;Check command reference number to see if it is the desired one.
674
675      000624' 200 01 1 00 000003 MSRV6: MOVE S1,@T1 ;Get cmd ref# from response
676      000625' 242 01 0 00 777774'      LSH S1,-4 ;Rt justify
677      000626' 316 01 0 00 005711'      CAMN S1,CMDREF ;Same as sent in command?
678      000627' 254 00 0 00 000633'      JRST MSRV8 ; Check returned status
679      $TEXT (<,<? MSGSRV - Command reference number from response not same as sent
680      000630' 260 17 0 00 000621*      Response = ^O/S1/, Sent = ^O/CMDREF/>)
681      000632' 254 00 0 00 000650'      JRST MSRV10 ;Re-queue buffer and exit
  
```

```

682
683 ;Check the returned status byte to see if it was what was expected.
684
685 000633' 260 17 0 00 001373' MSR8: $CALL CKSTAT ;Evaluate returned status byte
686 000634' 254 00 0 00 000650' JRST MSR10 ;Re-queue buffer and exit
687
688 ;Here if opcode 7 (OPCSDI), save command ref number and display returned data
689
690 000635' 200 01 1 00 000003 MSR7: MOVE S1,@T1 ;Get command ref number
691 000636' 202 01 0 00 005712' MOVEM S1,SDIREF ; and save it for use in SDI response
692 000637' 135 02 0 00 007025' LDB S2,[POINT 8,3(T1),7] ;Get byte count
693 000640' 200 06 0 00 007026' MOVE T4,[POINT 8,3(T1),15] ;Init output data pointer
694
695 000641' 361 02 0 00 000646' MSR2: SOJL S2,MSRV3 ;Any more characters? Jump if not
696 000642' 134 01 0 00 000006' ILDB S1,T4 ;Yes, put char in S1
697 000643' 322 01 0 00 000641' JUMPE S1,MSRV2 ;Ignore nulls
698 000644' 260 17 0 00 000225* $CALL BOUT% ;Output to terminal and log file
699 000645' 254 00 0 00 000641' JRST MSR2 ;Do next char
700
701 000646' 260 17 0 00 002213' MSR3: $CALL SENDIR ;Give response message
702 000647' 255 00 0 00 000000' JFCL ;Catch errors later
703
704 ;Here to requeue buffer when all done looking at this response.
705
706 000650' 332 00 0 00 005705' MSR10: SKIPE TRACE ;Trace enabled?
707 000651' 260 17 0 00 000630* $TEXT (,<MSGSRV - Attempting to re-queue buffer>)
708 000653' 260 17 0 00 002564' $CALL SSRM ;Re-queue buffer
709 000654' 255 00 0 00 000000' NOP ;Error ret
710
711 ;See if any further MESSAGE AVAILABLE before dismissing this interrupt
712
713 000655' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
714 000656' 260 17 0 00 000651* $TEXT (,<MSGSRV - Attempting to read status>)
715 000660' 260 17 0 00 002647' $CALL SSSTS ;Get status to see if more MSG AVAIL
716 000661' 254 00 0 00 000676' JRST MSRVX ;error, exit
717 000662' 200 01 0 00 002673' MOVE S1,STSARG+.SQFST ;Get ret flags,,connect state code
718 000663' 607 01 0 00 400000' ILNN S1,(SC%MSA) ;Message Available?
719 000664' 254 00 0 00 000676' JRST MSRVX ;no, exit
720 000665' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled
721 000666' 260 17 0 00 000656* $TEXT (,<MSGSRV - Another message available detected>)
722 000670' 254 00 0 00 000572' JRST MSR12 ;Repeat all until no more messages
723
724 ;Here if an error doing the SSRMG function, see if queue is empty.
725
726 000671' 200 02 0 00 005727' MSR16: MOVE S2,SCSERC ;Load error code
727 000672' 302 02 0 00 602613' CAIE S2,SCSQIE ;TOPS-20 receive queue empty?
728 000673' 306 02 0 00 000014' CAIN S2,SSRQEX ;TOPS-10 receive queue empty?
729 000674' 304 00 0 00 000000' CAIA ;Yes
730 000675' 254 00 0 00 003014' JRST FLAGIT ;Flag the error and return +1
731
732 000676' 476 00 0 00 005657' MSRVX: SETOM MAVAIL ;Set Msg Avail flag
733 000677' 263 17 0 00 000000' $RET ;Dismiss the interrupt

```

```

734
735      ; * CONNEC -- Request a connection with a node on the CI.
736      ;
737      ; The Node number has been obtained from the PARSER and stored in NODNAM.
738      ; Perform the connect request by the .SSCON SCS% JSYS.
739      ; The jsys will return as soon as the request has been sent.
740      ; Wait for the request to be processed via an EVENT interrupt.
741      ; Check the returned event code for Event Queue Empty and Connection Accepted.
742      ;
743      ;       Ret+1   if jsys failure or connection rejected
744      ;       Ret+2   if connection accepted
745
746      000700' 402 00 0 00 005711' CONNEC: SETZM   CMDREF      ;Init cmd ref#
747      000701' 260 17 0 00 001674'      $CALL   CLREFL      ;Clear all event flags
748      000702' 260 17 0 00 001704'      $CALL   BUFPTR      ;Reset all buffer pointers
749      000703' 332 00 0 00 005705'      $SKIPE  TRACE        ;Trace enabled?
750      000704' 260 17 0 00 000666*      $TEXT   (<CONNEC - Attempting a connection>)
751
752      000706' 260 17 0 00 002370'      $CALL   SSCON        ;Do CONNECT jsys
753      000707' 254 00 0 00 000730'      JRST    CONNX        ; error, exit
754
755      000710' 201 01 0 00 005670'      MOVEI   S1,^D3000      ;Set wait for 3 sec
756      000711' 260 17 0 00 001424'      $CALL   EVWAIT       ;Wait for event Q empty
757      000712' 254 00 0 00 000720'      JRST    CONN1        ; error ret
758
759      000713' 332 00 0 00 005654'      $SKIPE  CONACC       ;Queue empty, Connection Accepted?
760      000714' 254 00 0 00 000723'      JRST    CONN2        ; yes
761      000715' 250 17 0 00 000704*      $TEXT   (<? CONNEC - Connection not accepted...DUP Server off-line>)
762      000717' 254 00 0 00 000730'      JRST    CONNX
763
764      000720' 250 17 0 00 000715*      CONN1: $TEXT   (<? CONNEC - Could not detect Event Queue Empty by 3 sec>)
765      000722' 254 00 0 00 000730'      JRST    CONNX
766
767      000723' 332 00 0 00 005705'      CONN2: $SKIPE  TRACE        ;Trace enabled?
768      000724' 260 17 0 00 000720*      $TEXT   (<CONNEC - Connection Accepted...DUP Server on-line>)
769      000726' 260 17 0 00 003115'      $CALL   SPR1        ;Make spear entry
770      000727' 350 00 0 17 000000'      AOS      (P)        ;Ret+2
771      000730' 263 17 0 00 000000'      CONNX: $RET

```

```

772
773      ; * EMULAT - Send EXECUTE LOCAL PROGRAM to run local terminal emulator PSEUDO.
774      ;
775      ; Send the DUP 'EXECUTE LOCAL PROGRAM' packet to run PSEUDO, sending it as
776      ; the message portion of the .SSSMG jsys.
777      ; Wait for the request to be processed via an EVENT interrupt.
778      ; Check the returned event code for Event Queue Empty and MSG/DG Send Complete.
779      ;
780      ; Wait for a Message Available Interrupt and issue a .SSRMG jsys to read it.
781      ; (There is no event interrupt as a result of issuing .SSRMG).
782      ; This message is the response from the EXECUTE LOCAL PROGRAM command.
783      ;
784      ;       Ret+1   if jsys failure
785      ;       Ret+2   if successful
786
787      000731' 332 00 0 00 005705'      EMULAT: SKIPE   TRACE           ;Trace enabled?
788      000732' 260 17 0 00 000724*      $TEXT    (,<EMULAT - Attempting to execute local program>)
789      000734' 402 00 0 00 005655'      SETZM    SDONE           ;Clear some flags
790      000735' 402 00 0 00 005656'      SETZM    EQEMPTY
791      000736' 402 00 0 00 005657'      SETZM    MAVAIL
792
793      000737' 260 17 0 00 002100'      $CALL    EXLPGM          ;Execute local pgm (PSEUDO) via .SSSMG
794      000740' 254 00 0 00 001040'      JRST     EMLX             ; error, exit
795
796      000741' 201 01 0 00 005670'      MOVEI    S1,^D3000        ;Set wait for 3 sec
797      000742' 260 17 0 00 001424'      $CALL    EVWAIT          ;Wait for event Q empty
798      000743' 254 00 0 00 000751'      JRST     EML1             ; error ret
799
800      000744' 332 00 0 00 005655'      SKIPE    SDONE           ;Queue empty, Msg/DG send complete?
801      000745' 254 00 0 00 000754'      JRST     EML2             ; yes
802      000746' 260 17 0 00 000732*      $TEXT    (,<? EMULAT - Message send complete event not detected>)
803      000750' 254 00 0 00 001040'      JRST     EMLX
804
805      000751' 260 17 0 00 000746*      EML1:   $TEXT    (,<? EMULAT - Could not detect event queue empty by 3 sec>)
806      000753' 254 00 0 00 001040'      JRST     EMLX
807
808      000754' 332 00 0 00 005705'      EML2:   SKIPE    TRACE           ;Trace enabled?
809      000755' 260 17 0 00 000751*      $TEXT    (,<EMULAT - Message send complete>)
810
811      000757' 200 01 0 00 005665'      MOVE     S1,LPGTMO        ;Timeout (sec) value ret by DUST STATUS
812      000760' 260 17 0 00 001475'      $CALL    MAWAIT          ;Wait for message available interr
813      000761' 254 00 0 00 000763'      JRST     EML4             ; error ret
814      000762' 254 00 0 00 000766'      JRST     EML5             ;Msg available
815
816      000763'                                EML4:   $TEXT    (,<? EMULAT - No message available interrupt...exceeded timeout of ^D/LPGTMO
817      000763' 260 17 0 00 000755*      / sec>)
818      000765' 254 00 0 00 001040'      JRST     EMLX             ;exit

```

```

819
820 000766' 332 00 0 00 005662' EML5: SKIPE BADNOD ;Node# ret in RCV MSG ok?
821 000767' 254 00 0 00 001040' JRST EMLX ; no
822
823 ; Evaluate the response returned from EXECUTE LOCAL PROGRAM Command.
824
825 000770' 402 00 0 00 005675' SETZM STANDA ;Init flags
826 000771' 402 00 0 00 005676' SETZM OVERLA
827 000772' 402 00 0 00 005677' SETZM WRITEB
828 000773' 402 00 0 00 005700' SETZM STDIAL
829
830 000774' 200 03 0 00 005703' MOVE T1,BUFADR ;BUFADR previously loaded in MSGSRV
831 000775' 135 01 0 00 007275' LDB S1,[POINT 8,3(T1),31] ;Get flags byte
832 000776' 322 00 0 00 001027' JUMPE EML13 ;Jump if flags = 0
833 000777' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
834 001000' 260 17 0 00 000763* $TEXT (<CKLOCP - Execute local program response flags>)
835 001002' 606 01 0 00 000001' TRNN S1,1 ;Stand-Alone program?
836 001003' 254 00 0 00 001007' JRST EML7 ; no
837 001004' 476 00 0 00 005675' SETOM STANDA ;Yes
838 001005' 260 17 0 00 001000* $TEXT (<% WARNING: This is a Stand-Alone program>)
839
840 001007' 606 01 0 00 000002' EML7: TRNN S1,2 ;Program require overlays from host?
841 001010' 254 00 0 00 001014' JRST EML9 ; no
842 001011' 476 00 0 00 005676' SETOM OVERLA ;Yes
843 001012' 260 17 0 00 001005* $TEXT (<% WARNING: This program requires overlay buffers from host>)
844
845 001014' 606 01 0 00 000004' EML9: TRNN S1,4 ;Should overlays be writeable?
846 001015' 254 00 0 00 001021' JRST EML11 ; no
847 001016' 476 00 0 00 005677' SETOM WRITEB ;yes
848 001017' 260 17 0 00 001012* $TEXT (< Overlays should be writeable from the controller>)
849
850 001021' 606 01 0 00 000010' EML11: TRNN S1,10 ;Program use standard DUP dialogue?
851 001022' 254 00 0 00 001027' JRST EML13 ; no
852 001023' 476 00 0 00 005700' SETOM STDIAL ;Yes
853 001024' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
854 001025' 260 17 0 00 001017* $TEXT (< This program uses standard DUP dialogue>)
855
856 001027' 135 01 0 00 007404' EML13: LDB S1,[POINT 16,3(T1),15] ;Get pgm version bytes
857 001030' 202 01 0 00 005673' MOVEM S1,PGMVER ; and save it
858
859 001031' 135 01 0 00 007405' LDB S1,[POINT 8,3(T1),23] ;Get timeout value byte
860 001032' 202 01 0 00 005674' MOVEM S1,DATTMO ; and save it
861
862 001033' 336 00 0 00 005705' SKIPN TRACE ;Trace enabled?
863 001034' 254 00 0 00 001037' JRST EML15 ; no
864 $TEXT (< Program version = ^D/PGMVER/
865 001035' 260 17 0 00 001025* Timeout value for SEND-RCV DATA commands = ^D/DATTMO/ SEC>)
866 001037' 350 00 0 17 000000 EML15: AOS (P)
867 001040' 263 17 0 00 000000 EMLX: $RET

```

```

868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885 001041' 304 00 0 00 000000
886 001047' 332 00 0 00 005705'
887 001050' 260 17 0 00 001035*
888 001052' 402 00 0 00 005655'
889 001053' 402 00 0 00 005656'
890 001054' 402 00 0 00 005657'
891
892
893
894 001055' 200 04 0 00 007473'
895 001056' 200 05 0 00 006016'
896 001057' 200 01 0 00 005714'
897 001060' 136 01 0 00 000004
898 001061' 136 00 0 00 000004
899
900 001062' 134 01 0 00 000005
901 001063' 136 01 0 00 000004
902 001064' 332 00 0 00 000000*
903 001065' 260 17 0 00 000000*
904 001066' 332 00 0 00 005705'
905 001067' 260 17 0 00 001050*
906 001071' 372 00 0 00 005714'
907 001072' 254 00 0 00 001062'

; * SNDHSC - Send a SEND DATA IMMEDIATE packet to the HSC50
; Enter with characters in TTIBUF and IBUFCT to the number of chars to send.
; Load the arg block of the SSSMG jsys with the SDI opcode and data.
; Send the DUP - SEND DATA IMMED as the msg portion of the .SSSMG jsys.
; Wait for the request to be processed via an EVENT interrupt.
; Check the returned event code for Event Queue Empty and MSG/DG Send Complete.
; Wait for a Message Available Interrupt and issue a .SSRMG jsys to read it.
; (There is no event interrupt as a result of issuing .SSRMG).
; This message is the response from the SEND DATA command.
;
; Ret+1 if jsys failure
; Ret+2 if successful

SNDHSC: $SAVE <S1,S2,T2>
        SKIPE TRACE ;Trace enabled?
        $TEXT (,<SNDHSC - Attempting to Send Data Immed to the HSC of ^D/IBUFCT/ bytes:>)
        SETZM SDONE ;Clear some flags
        SETZM EQEMTY
        SETZM MAVAIL

;Copy terminal input buffer to send block.

        MOVE T2,[POINT 8,SDCBLK+3] ;Init arg block pointer
        MOVE T3,[POINT 7,TTIBUF] ; and TTIBUF
        MOVE S1,IBUFCT
        IDPB S1,T2 ;Load char count in arg block
        IBP T2 ;Skip over 2nd byte (always 0)

SND1: ILDB S1,T3 ;Get byte from TTIBUF
      IDPB S1,T2 ; and put in arg block
      SKIPE LOGIFN ;Skip if not logging
      $CALL LOUT%% ;Yep, copy to file
      SKIPE TRACE ;Trace mode?
      $TEXT (,<^O/S1/ ^A>) ;Yes output the byte
      SOSE IBUFCT ;All chars loaded?
      JRST SND1 ; no

```

```

908
909 001073' 260 17 0 00 002160'          $CALL SENDIC          ;Do Send Data Immed to HSC via .SSSMG
910 001074' 254 00 0 00 001125'          JRST SDATX          ; error, exit
911
912                                     ;Wait for message send complete and return.
913
914 001075' 201 01 0 00 005670'          MOVEI S1,^D3000          ;Set wait for 3 sec
915 001076' 260 17 0 00 001424'          $CALL EVWAIT          ;Wait for event Q empty
916 001077' 254 00 0 00 001105'          JRST SDAT1          ; error ret
917
918 001100' 332 00 0 00 005655'          SKIPE SDONE          ;Queue empty, Msg/DG send complete?
919 001101' 254 00 0 00 001110'          JRST SDAT2          ; yes
920 001102' 260 17 0 00 001067*          $TEXT (,<? SNDDAT - Message send complete event not detected>)
921 001104' 254 00 0 00 001125'          JRST SDATX
922
923 001105' 260 17 0 00 001102*          SDAT1: $TEXT (,<? SNDDAT - Could not detect event queue empty by 3 sec>)
924 001107' 254 00 0 00 001125'          JRST SDATX
925
926 001110' 332 00 0 00 005705'          SDAT2: SKIPE TRACE          ;Trace enabled?
927 001111' 260 17 0 00 001105*          $TEXT (,<SNDDAT - Message send complete>)
928
929 001113' 200 01 0 00 005674'          MOVE S1,DATTMO          ;Timeout (sec) ret by EX LOCAL PGM cmd
930 001114' 260 17 0 00 001475'          $CALL MAWAIT          ;Wait for message available interr
931 001115' 254 00 0 00 001117'          JRST SDAT4          ; error ret
932 001116' 254 00 0 00 001122'          JRST SDAT5          ;Msg available
933
934 001117'                                SDAT4: $TEXT (,<? SNDDAT - No message available interrupt...exceeded timeout of ^D/DATTMO
935 001117' 260 17 0 00 001111*          / sec>)
936 001121' 254 00 0 00 001125'          JRST SDATX          ;exit
937
938 001122' 332 00 0 00 005662'          SDAT5: SKIPE BADNOD          ;Node# ret in RCV MSG ok?
939 001123' 254 00 0 00 001125'          JRST SDATX          ; no, exit
940 001124' 350 00 0 17 000000          AOS (P)          ;Ret+2
941 001125' 263 17 0 00 000000          SDATX: $RET

```

SUBTTL ABORT

* ABORTT -- Issue the ABORT DUP command to terminate a program/go back to IDLE
 Build the "ABORT" packet as the message portion of the .SSSMG jsys.
 Wait for the request to be processed via an EVENT interrupt.
 Check the returned event code for Event Queue Empty and MSG/DG Send Complete.
 Wait for a Message Available Interrupt and issue a .SSRMG jsys to read it.
 (There is no event interrupt as a result of issuing .SSRMG).
 This message is the response from the ABORT command.

```

Ret+1  if jsys failure
Ret+2  if successful

ABORTT: SKIPE  TRACE           ;Trace enabled?
        $TEXT  (<ABORTT - Attempting to do the ABORT command>)
        SETZM  SDONE          ;Clear some flags
        SETZM  EQEMPTY
        SETZM  MAVAIL

        $CALL  ABORT           ;Send DUP - ABORT command (via .SSSMG)
        JRST  ABTX            ; error, exit

        MOVEI  S1,^D10000     ;Set wait for 10 sec
        $CALL  EVWAIT         ;Wait for event Q empty
        JRST  ABT1            ; error ret
        SKIPE  SDONE          ;Queue empty, Msg/DG send complete?
        JRST  ABT2            ; yes
        $TEXT  (<? ABORTT - Message send complete event not detected>)
        JRST  ABTX

ABT1:   $TEXT  (<? ABORTT - Could not detect event queue empty by 10 sec>)
        JRST  ABTX

ABT2:   SKIPE  TRACE           ;Trace enabled?
        $TEXT  (<ABORTT - Message send complete>)
        MOVEI  S1,^D10        ;Set wait for 10 sec
        $CALL  MAWAIT         ;Wait for message available interr
        JRST  ABT4            ; error ret
        JRST  ABT5            ;Msg available

ABT4:   $TEXT  (<? ABORTT - No message available interrupt by 10 sec>)
        JRST  ABTX            ;exit

ABT5:   SKIPE  BADNOD          ;Node# ret in RCV MSG ok?
        JRST  ABTX            ; no
        AOS    (P)            ;Ret+2

ABTX:   $RET

```

```

942
943
944
945
946
947
948
949
950
951
952
953
954
955
956 001126' 332 00 0 00 005705'
957 001127' 260 17 0 00 001117*
958 001131' 402 00 0 00 005655'
959 001132' 402 00 0 00 005656'
960 001133' 402 00 0 00 005657'
961
962 001134' 260 17 0 00 002236'
963 001135' 254 00 0 00 001166'
964
965 001136' 201 01 0 00 023420
966 001137' 260 17 0 00 001424'
967 001140' 254 00 0 00 001146'
968 001141' 332 00 0 00 005655'
969 001142' 254 00 0 00 001151'
970 001143' 260 17 0 00 001127*
971 001145' 254 00 0 00 001166'
972
973 001146' 260 17 0 00 001143*
974 001150' 254 00 0 00 001166'
975
976 001151' 332 00 0 00 005705'
977 001152' 260 17 0 00 001146*
978 001154' 201 01 0 00 000012
979 001155' 260 17 0 00 001475'
980 001156' 254 00 0 00 001160'
981 001157' 254 00 0 00 001163'
982
983 001160' 260 17 0 00 001152*
984 001162' 254 00 0 00 001166'
985
986 001163' 332 00 0 00 005662'
987 001164' 254 00 0 00 001166'
988 001165' 350 00 0 17 000000
989 001166' 263 17 0 00 000000

```

```

990          SUBTTL Disconnect
991
992          ;* DISCON -- Disconnect and close the Connection previously made.
993
994          ; Perform the disconnect request by the .SSDIS SCS% jsys.
995          ; There will be no EVENT interrupt as a result of doing the .SSDIS
996          ; Verify the connection is closed by doing the SSSTS SCS% jsys to obtain status
997          ; about the connection. An Invalid CID may be an expected error. Ignore it.
998
999          ;
1000          Ret+1 always
1001
1002          DISCON: $SAVE <S1,S2>
1003          $SKIPE TRACE ;Trace enabled?
1004          $TEXT (,<DISCON - Attempting to disconnect>)
1005
1006          $CALL SSDIS ;Do DISCONNECT jsys
1007          JRST DSC4 ; error, exit
1008
1009          $CALL SPR2 ;Write to spear if needed
1010          $SKIPE TRACE ;Trace?
1011          $TEXT (,<DISCON - Connection closed>)
1012          $RET
1013
1014          DSC4: $CALL FLAGIT ;Error doing the disconnect
1015          $TEXT (,<? DISCON - Disconnect failed>)
1016          $RET
  
```



```

1048
1049 001234' 260 17 0 00 001231* DUST1: $TEXT (,<? DUSTAT - Could not detect event queue empty by 3 sec>)
1050 001236' 254 00 0 00 001253' JRST DUSTX
1051
1052 001237' 332 00 0 00 005705' DUST2: SKIPE TRACE ;Trace enabled?
1053 001240' 260 17 0 00 001234* $TEXT (,<DUSTAT - Message send complete>)
1054 001242' 201 01 0 00 000060 MOVEI SI,60 ;Set wait for 60 sec
1055 001243' 260 17 0 00 001475' $CALL MAWAIT ;Wait for message available interr
1056 001244' 254 00 0 00 001246' JRST DUST4 ;error ret
1057 001245' 254 00 0 00 001251' JRST DUST5 ;Msg available
1058
1059 001246' 260 17 0 00 001240* DUST4: $TEXT (,<? DUSTAT - No message available interrupt by 60 sec>)
1060 001250' 254 00 0 00 001253' JRST DUSTX ;exit
1061
1062 001251' 336 00 0 00 005662' DUST5: SKIPN BADNOD ;Node# ret in RCV MSG ok?
1063 001252' 350 00 0 17 000000 AOS (P) ;No, Ret+2
1064 001253' 263 17 0 00 000000 DUSTX: $RET
  
```

1065
 1066
 1067
 1068
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079
 1080
 1081 001254' 304 00 0 00 000000
 1082 001263' 402 00 0 00 005666'
 1083 001264' 402 00 0 00 005667'
 1084 001265' 402 00 0 00 005670'
 1085 001266' 402 00 0 00 005671'
 1086 001267' 402 00 0 00 005672'
 1087
 1088 001270' 332 00 0 00 005705'
 1089 001271' 260 17 0 00 001246*
 1090
 1091 001273' 200 03 0 00 005703'
 1092 001274' 135 01 0 00 007275'
 1093 001275' 606 01 0 00 000001
 1094 001276' 254 00 0 00 001302'
 1095 001277' 472 00 0 00 005666'
 1096
 1097 001300' 260 17 0 00 001271*
 1098
 1099 001302' 602 01 0 00 000002
 1100 001303' 254 00 0 00 001307'
 1101
 1102 001304' 260 17 0 00 001300*
 1103 001306' 254 00 0 00 001314'
 1104
 1105 001307' 476 00 0 00 005667'
 1106 001310' 336 00 0 00 005705'
 1107 001311' 254 00 0 00 001314'
 1108 001312' 260 17 0 00 001304*
 1109
 1110 001314' 606 01 0 00 000004
 1111 001315' 254 00 0 00 001322'
 1112 001316' 476 00 0 00 005670'
 1113 001317' 332 00 0 00 005705'
 1114 001320' 260 17 0 00 001312*
 1115

* CKDUST - Evaluate the response message returned from the GET DUST STATUS msg
 Evaluate returned FLAGS and set the following registers:
 DISSRV to -1 if any program execution DISABLES operation of all other
 servers in the same controller.
 LODMED to -1 if controller HAS local load media for loading diags/utilities
 NOEXSP to -1 if server will NOT accept the Execute Supplied Program command
 ACTIVE to -1 if server is currently active
 SNDDIM to -1 if server has Send Data Immediate support.
 Load PRGIND with progress indicator for running remote programs
 Load LPGTMO with timeout value (in sec) to use for Execute Local Program Cmd.
 Used only if LODMED above is set.
 CKDUST: \$SAVE <S1,S2,T1,T2>
 SETZM DISSRV ;Init flags
 SETZM LODMED
 SETZM NOEXSP
 SETZM ACTIVE
 SETZM SNDDIM
 SKIPE TRACE ;Trace enabled?
 \$TEXT (,<CKDUST - DUST response flags: ^A>)
 MOVE T1,BUFADR ;BUFADR previously loaded by MSGSRV
 LDB S1,[POINT 8,3(T1),31] ;Get flags byte
 TRNN S1,1 ;Disable other servers?
 JRST CKD3 ;no
 SETOM DISSRV ;Yes
 \$TEXT (,<? Any pgm execution under this server will disable operation of all
 other servers in this controller.>)
 CKD3: TRNE S1,2 ;Does controller have loc load media?
 JRST CKD4 ;yes
 \$TEXT (,<? HSC does not have local load media for loading diags and utilities
 cannot connect>)
 JRST CKD5
 CKD4: SETOM LODMED
 SKIPN TRACE ;Trace enabled?
 JRST CKD5 ;no
 \$TEXT (,< Controller has local load media>)
 CKD5: TRNN S1,4 ;Server accept EXE SUPPLIED PGM cmd?
 JRST CKD6 ;yes
 SETOM NOEXSP ;no
 SKIPE TRACE
 \$TEXT (,< This server will not accept the Execute Supplied Pgm Command>)

```

1116
1117 001322' 606 01 0 00 000010 CKD6: TRNN S1,10 ;Server currently ACTIVE?
1118 001323' 254 00 0 00 001331' JRST CKD7 ; no
1119 001324' 476 00 0 00 005671' SETOM ACTIVE ;Yes
1120 001325' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
1121 001326' 260 17 0 00 001320* $TEXT (,< Server currently ACTIVE>)
1122 001330' 254 00 0 00 001334' JRST CKD8
1123
1124 001331' 332 00 0 00 005705' CKD7: SKIPE TRACE ;Trace enabled?
1125 001332' 260 17 0 00 001326* $TEXT (,< Server currently IDLE>)
1126
1127 001334' 606 01 0 00 000020 CKD8: TRNN S1,20 ;Send Data Immed support?
1128 001335' 254 00 0 00 001343' JRST CKD9 ; no
1129 001336' 476 00 0 00 005672' SETOM SMDDIM ;Yes
1130 001337' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
1131 001340' 260 17 0 00 001332* $TEXT (,< DUP supports Send Data Immediate>)
1132 001342' 254 00 0 00 001345' JRST CKD10
1133
1134 001343' 260 17 0 00 001340* CKD9: $TEXT (,<? DUP Does not support Send Data Immediate, cannot connect>)
1135
1136 001345' 135 01 0 00 010266' CKD10: LDB S1,[POINT 24,3(T1),23] ;Get pgm ext bytes
1137 001346' 202 01 0 00 005663' MOVEM S1,PGMEXT ; and save it
1138
1139 001347' 135 01 0 00 010267' LDB S1,[POINT 8,4(T1),7] ;Prog indicator byte 0
1140 001350' 137 01 0 00 010270' DPB S1,[POINT 8,PRGIND,35] ; to LSB
1141 001351' 135 01 0 00 010271' LDB S1,[POINT 8,4(T1),15] ;Prog indicator byte 1
1142 001352' 137 01 0 00 010272' DPB S1,[POINT 8,PRGIND,27]
1143 001353' 135 01 0 00 010273' LDB S1,[POINT 8,4(T1),23] ;Prog indicator byte 2
1144 001354' 137 01 0 00 010274' DPB S1,[POINT 8,PRGIND,19]
1145 001355' 135 01 0 00 010275' LDB S1,[POINT 8,4(T1),31] ;Prog indicator byte 3
1146 001356' 137 01 0 00 010276' DPB S1,[POINT 8,PRGIND,11] ; to MSB
1147
1148 001357' 135 01 0 00 010277' LDB S1,[POINT 8,5(T1),7] ;Timeout value byte 0
1149 001360' 137 01 0 00 010300' DPB S1,[POINT 8,LPGTMO,35] ; to LSB
1150 001361' 135 01 0 00 010301' LDB S1,[POINT 8,5(T1),15] ;Timeout value byte 1
1151 001362' 137 01 0 00 010302' DPB S1,[POINT 8,LPGTMO,27] ; to MSB
1152
1153 001363' 336 00 0 00 005705' SKIPN TRACE ;Trace enabled?
1154 001364' 254 00 0 00 001372' JRST CKDX ; no, exit
1155 001365' 135 01 0 00 007025' LDB S1,[POINT 8,3(T1),7] ;Byte 0
1156 001366' 135 02 0 00 007026' LDB S2,[POINT 8,3(T1),15] ;Byte 1
1157 001367' 135 04 0 00 007405' LDB T2,[POINT 8,3(T1),23] ;Byte 2
1158 $TEXT (,< Program extension = ^7/S1/^7/S2/^7/T2/
1159 Progress indicator = ^D/PRGIND/
1160 Timeout value for Execute Local Program commands = ^D/LPGTMO/ sec>)
1161
1162 001372' 263 17 0 00 000000 CKDX: $RET

```

SUBTTL Support Subroutines

;* CKSTAT - Check the status byte of a DUP response message.

```

: GET DUST STATUS: 0 = Success
: EXECUTE SUPPLIED PGM: 0 = Success
:                      1 = Invalid cmd: Server not in IDLE state/cmd fld error
:                      2 = No region available
:                      3 = No region suitable
:                      9 = Host buffer access error: Unibus error or
:                          invalid byte count
: EXECTUE LOCAL PGM: 0-3 = same as above
:                      4 = Program not known: no such program on media
:                      5 = Load failure: input error while loading program
:                      6 = Standalone: SA modifier not specified for a SA prog
: SEND/RCV DATA: 0 = Success
:                  1 = Invalid cmd: Server not in ACTIVE state/cmd fld error
: SEND DATA IMMEDIATE: (same as send/rec data)
: ABORT PROGRAM: (same as send/rec data)

```

```

CKSTAT: $SAVE <S1,S2,T1>
        MOVE T1, BUFADR ;BUFADR previously loaded in MSGSRV
        LDB S1,[POINT 7,2(T1),23] ;Get status byte 0
        SKIPE S1 ;Zero?
        JRST CKST1 ;no, print it
        SKIPE TRACE ;Yes
        $TEXT (<CKSTAT - Response status successful>)
        JRST CKSTX

```

```

CKST1: CAILE S1,^D9 ;Less or equal to max?
        MOVEI S1,^D0 ;no, illegal error code
        MOVE S2,[
          [ASCII/Invalid error code/]
          [ASCII/Invalid command/]
          [ASCII/No memory available/]
          [ASCII/No memory suitable/]
          [ASCII/Program not known/]
          [ASCII/Load failure/]
          [ASCII/Standalone/]
          [ASCII/Unknown status code = 7/]
          [ASCII/Unknown status code = 8/]
          [ASCII/Host buffer access error/]](S1)

```

```

        $TEXT (<? CKSTAT - Response status unsuccessful
Code ^D/S1/: ^T/(S2)/^A>)
        LDB S2,[POINT 7,2(T1),31] ;Get offset
        CAIN S1,1 ;Is it INVALID COMMAND?
        $TEXT (< offset ^0/S1/^A>) ;Yes
        $CALL PCRLF ;Output CRLF
CKSTX: $RET

```

```

1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183 001373' 304 00 0 00 000000
1184 001401' 200 03 0 00 005703'
1185 001402' 135 01 0 00 010351'
1186 001403' 332 00 0 00 000001
1187 001404' 254 00 0 00 001411'
1188 001405' 332 00 0 00 005705'
1189 001406' 260 17 0 00 001370*
1190 001410' 254 00 0 00 001423'
1191
1192 001411' 303 01 0 00 000011
1193 001412' 201 01 0 00 000000
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203 001413' 200 02 0 01 010437'
1204
1205 001414' 260 17 0 00 001406*
1206 001416' 135 02 0 00 010475'
1207 001417' 306 01 0 00 000001
1208 001420' 260 17 0 00 001414*
1209 001422' 260 17 0 00 000220*
1210 001423' 263 17 0 00 000000

```

```

1211
1212      ;* EVWAIT -- Wait a specified amount of time for Event Queue Empty
1213      ;
1214      ; Exit before specified time if EQEMPTY sets from the EVENT Interrupt Handler
1215      ; Enter with the specified time in S1 (ms) and save it in DLYTIM (ms).
1216      ;
1217      ;      Ret+1   if Event Queue not empty by the specified time: EQEMPTY= 0
1218      ;      Ret+2   if Event Queue Empty by specified time:      EQEMPTY= -1
1219
1220      001424' 304 00 0 00 000000      EVWAIT: $SAVE    <S1,S2>      ;Save some ac's
1221      001431' 202 01 0 00 005706'      MOVEM     S1,DLYTIM      ;Save delay time in ms
1222
1223      001432' 260 17 0 00 001444'      $CALL     GETTIM      ;Get current time from monitor in ms
1224      001433' 202 01 0 00 005707'      MOVEM     S1,DSTRT      ;Save the starting time
1225
1226      001434' 260 17 0 00 001444'      EVW2:  $CALL     GETTIM      ;Get the current time from monitor
1227      001435' 274 01 0 00 005707'      SUB       S1,DSTRT      ;Calculate elapsed time
1228      001436' 313 01 0 00 005706'      CAMLE     S1,DLYTIM      ;Have delayed long enough ?
1229      001437' 254 00 0 00 001443'      JRST      EVWX          ; Yes, exit
1230      001440' 336 00 0 00 005656'      SKIPN    EQEMPTY      ;No, event Q empty?
1231      001441' 254 00 0 00 001434'      JRST      EVW2          ; no, try again
1232      001442' 350 00 0 17 000000      AOS       (P)          ;Yes, ret+2
1233      001443' 263 17 0 00 000000      EVWX:  $RET
  
```

```

1234
1235      ;* GETTIM - Read current system runtime into S1
1236
1237 001444' 304 00 0 00 000000 GETTIM: $SAVE <S2> ;Save S2 (smashed by TIME% and IDIVI)
1238 001450' 336 00 0 00 000342* SKIPN MONTYP ;TOPS-10?
1239 001451' 254 00 0 00 001454' JRST GETTI1 ;Yes
1240 001452' 104 00 0 00 000014 TIME% ;The JSYS to get system uptime in MS
1241 001453' 263 17 0 00 000000 $RET
1242
1243 001454' 336 00 0 00 001474' GETTI1: SKIPN CYCL60 ;Computed 60HZ flag yet?
1244 001455' 260 17 0 00 001464' $CALL GETTI2 ;No, get it now
1245 001456' 200 01 0 00 010514' MOVE S1,[136,,11] ;System uptime in jiffies %CNSUP
1246 001457' 047 01 0 00 000041 GETTAB S1, ;or CALLI S1,41
1247 001460' 400 01 0 00 000000 SETZ S1, ;not likely it will fail
1248 001461' 221 01 0 00 001750 IMULI S1,*D1000 ;convert to milli-jiffies
1249 001462' 230 01 0 00 001474' IDIV S1,CYCL60 ;Convert to milli-seconds
1250 001463' 263 17 0 00 000000 $RET ;Return
1251
1252 001464' 200 01 0 00 010515' GETTI2: MOVE S1,[17,,11] ;Load %CNSTS system flags
1253 001465' 047 01 0 00 000041 GETTAB S1, ;or CALLI S1,41
1254 001466' 400 01 0 00 000000 SETZ S1, ;should not execute this, assume 60
1255 001467' 603 01 0 00 004000 TLNE S1,(1B6) ;Is ST%CYC set for 50Hz?
1256 001470' 334 01 0 00 010516' SKIPN S1,[*D50] ;Yes, this must be Europe mon ami
1257 001471' 201 01 0 00 000074 MOVEI S1,*D60 ;No, this is anywhere else
1258 001472' 202 01 0 00 001474' MOVEM S1,CYCL60 ;Set divisor
1259 001473' 263 17 0 00 000000 $RET ; and return
1260
1261 001474' CYCL60: BLOCK 1 ;60. or 50. jiffies per second

```

```

1262
1263      ; * MAWAIT -- Wait a specified amount of time for Message Available Interrupt
1264      ;
1265      ; Exit before specified time if MAVAIL sets from the MSGSRV Interrupt Handler
1266      ; Enter with the specified time in S1 (sec) and save it in DLYTIM (in ms).
1267      ;
1268      ;      Ret+1   if Message Not Available by the specified time: MAVAIL= 0
1269      ;      Ret+2   if Message Available by specified time:      MAVAIL= -1
1270
1271      001475' 304 00 0 00 000000      MAWAIT: $SAVE    <S1,S2>      ;Save some ac's
1272      001502' 336 00 0 00 000001      SKIPN    S1          ;Timeout zero?
1273      001503' 201 01 0 00 000036      MOVEI    S1,^D30      ; Yes, default to 30 sec
1274      001504' 221 01 0 00 001750      IMULI    S1,^D1000    ;in ms
1275      001505' 202 01 0 00 005706'      MOVEM    S1,DLYTIM    ;Save it
1276
1277      001506' 260 17 0 00 001444'      $CALL    GETTIM      ;Get current time from monitor in ms
1278      001507' 202 01 0 00 005707'      MOVEM    S1,DSTRT    ;Save the starting time
1279
1280      001510' 260 17 0 00 001444'      MAW2:   $CALL    GETTIM      ;Get the current time from monitor
1281      001511' 274 01 0 00 005707'      SUB      S1,DSTRT    ;Calculate elapsed time
1282      001512' 313 01 0 00 005706'      CAMLE    S1,DLYTIM    ;Have delayed long enough ?
1283      001513' 254 00 0 00 001517'      JRST     MAWX        ; Yes, exit
1284      001514' 336 00 0 00 005657'      SKIPN    MAVAIL      ;No, Msg Avail?
1285      001515' 254 00 0 00 001510'      JRST     MAW2        ; no, try again
1286      001516' 350 00 0 17 000000      AOS      (P)         ;Yes, ret+2
1287      001517' 263 17 0 00 000000      MAWX:   $RET

```

```

1288
1289      ;* INIT1 - Initialize buffers, flags, timer and setup terminal
1290
1291      001520' 304 00 0 00 000000      INIT1: $SAVE    <S1,S2,T3>
1292      001526' 402 00 0 00 005714'      SETZM      IBUFCT      ;Init terminal input buffer char ctr
1293
1294      001527' 402 00 0 00 005713'      SETZM      DISFLG      ;Disconnect character received flag
1295
1296      001530' 260 17 0 00 000334'      $CALL      INITIM      ;Init reservation timer
1297      001531' 336 00 0 00 001450*      SKIPN      MONTYP      ;TOPS-20?
1298      001532' 263 17 0 00 000000      $RET          ;Nope, TOPS-10, return now
1299
1300      001533' 201 01 0 00 000100      MOVEI      S1,.PRIIN      ;Specified terminal
1301      001534' 104 00 0 00 000107      RFMOD%      ;Read its JFN mode word
1302      001535' 202 02 0 00 005722'      MOVEM      S2,ITMOD      ; and save it (restored in RESTOR)
1303      001536' 620 02 0 00 000300      TRZ        S2,3B29      ;change to: 7 bit in/out
1304
1305      001537' 104 00 0 00 000110      SFMOD%      ;no format control, no translation
1306
1307      001540' 201 01 0 00 400000      MOVEI      S1,.FHSLF      ;Set program related modes for
1308      001541' 104 00 0 00 000173      RTIW%      ; specified terminal
1309      001542' 202 02 0 00 005723'      MOVEM      S2,TTIWD      ;Process handle
1310      001543' 201 01 0 00 400000      MOVEI      S1,.FHSLF      ;Read terminal interrupt word
1311      001544' 201 02 0 00 000020      MOVX      S2,1B<.TICTI> ; and save it (restored in RESTOR)
1312      001545' 104 00 0 00 000174      STIW%      ;Process handle
1313
1314      001546' 263 17 0 00 000000      $RET          ;All type in
                                ;Set modified terminal interrupt word

```

```

1315
1316      ;* SYSINT - Enable system interrupt channels
1317
1318      001547' 304 00 0 00 000000      SYSINT: $SAVE    <S1,S2>
1319      001554' 336 00 0 00 001531*      SKIPN    MONTP    ;Skip if TOPS-20
1320      001555' 254 00 0 00 001566'      JRST     SYSIN1   ;TOPS-10
1321      001556' 205 01 0 00 000037      MOVX     S1,<.TICTI,,TTICHN> ;Assign terminal interr for all type in
1322      001557' 104 00 0 00 000137      ATIX%                   ; for ch 0
1323      001560' 200 01 0 00 010526'      MOVX     S1,<.TICCC,,CCCCHN> ;Assign control-C
1324      001561' 104 00 0 00 000137      ATIX%                   ; interrupts to that channel
1325      001562' 201 01 0 00 400000      MOVEI    S1,.FHSLF    ;Current process handle
1326      001563' 200 02 0 00 010527'      MOVX     S2,ONCHNS   ;Activate all channels
1327      001564' 104 00 0 00 000131      AIC%      ;Acitivate Interrupt Channels
1328      001565' 254 00 0 00 000000*      JRST     I%ION      ;Enable interrupts and return to caller
1329
1330      ;Here if TOPS-10 to set up interrupt system
1331
1332      001566' 200 01 0 00 010533'      SYSIN1: MOVE    S1,[PS.FAC![EXP .PCSCS,<SCOFFS,,0>,<2,,0>]]
1333      001567' 047 01 0 00 000136      PISYS.    S1,        ;Enable the channel for SCS ints
1334      001570' 260 17 0 00 001420*      $TEXT    (,<? Can't activate interrupts for SCS - error code ^0/S1/>)
1335      001572' 200 01 0 00 010557'      MOVE     S1,[PS.FAC![EXP .PCTMR,<TIOFFS,,0>,<1,,0>]]
1336      001573' 047 01 0 00 000136      PISYS.    S1,        ;Enable the channel for PITMR ints
1337      001574' 260 17 0 00 001570*      $TEXT    (,<? Can't activate interrupts for PITMR - error code ^0/S1/>)
1338      001576' 200 01 0 00 010604'      MOVE     S1,[PS.FAC![EXP .PCJBI,<TTOFFS,,0>,<1,,0>]]
1339      001577' 047 01 0 00 000136      PISYS.    S1,        ;Enable the channel for self made ints
1340      001600' 260 17 0 00 001574*      $TEXT    (,<? Can't activate interrupts for PIJBI - error code ^0/S1/>)
1341      001602' 200 01 0 00 010626'      MOVE     S1,[PS.FAC![EXP <SIXBIT/TTY/>,<TTOFFS,,PS.RID!PS.RIA>,<1,,0>]]
1342      001603' 047 01 0 00 000136      PISYS.    S1,        ;Enable interrupts for terminal chars
1343      001604' 260 17 0 00 001600*      $TEXT    (,<? Can't activate TTY interrupts - error code ^0/S1/>)
1344      001606' 200 01 0 00 010652'      MOVE     S1,[PS.FAC![EXP .PCSTP,<CCOFFS,,0>,<3,,0>]]
1345      001607' 047 01 0 00 000136      PISYS.    S1,        ;Enable interrupts for control-C
1346      001610' 260 17 0 00 001604*      $TEXT    (,<? Can't activate control-C interrupts - error code ^0/S1/>)
1347      001612' 254 00 0 00 001565*      JRST     I%ION      ;Turn on interrupt system

```

```

1348
1349      ;* RESTOR - Deactivate interrupts and restore terminal to original mode word
1350
1351 001613' 304 00 0 00 000000 RESTOR: $SAVE <S1,S2>
1352 001620' 260 17 0 00 000000* $CALL IXIOFF ;Turn off interrupts
1353 001621' 332 00 0 00 001554* SKIPE MONTYP ;Skip if TOPS-10
1354 001622' 254 00 0 00 001650' JRST RESTO2 ;TOPS-20
1355
1356      ;Remove TOPS-10 interrupt system
1357
1358 001623' 200 01 0 00 010674' MOVE S1,[PS.FRC![EXP .PCSCS,<SCOFFS,,0>,<2,,0>]]
1359 001624' 047 01 0 00 000136 PISYS. S1, ;Disable SCA ints
1360 001625' 260 17 0 00 001610* $TEXT (,<? Can't remove interrupts for SCS - error code ^0/S1/>)
1361 001627' 200 01 0 00 010715' MOVE S1,[PS.FRC![EXP .PCTMR,<TIOFFS,,0>,<1,,0>]]
1362 001630' 047 01 0 00 000136 PISYS. S1, ;Disable PITMR ints
1363 001631' 260 17 0 00 001625* $TEXT (,<? Can't remove interrupts for PITMR - error code ^0/S1/>)
1364 001633' 200 01 0 00 010736' MOVE S1,[PS.FRC![EXP .PCJBI,<TIOFFS,,0>,<1,,0>]]
1365 001634' 047 01 0 00 000136 PISYS. S1, ;Disable self made ints
1366 001635' 260 17 0 00 001631* $TEXT (,<? Can't remove interrupts for PIJBI - error code ^0/S1/>)
1367 001637' 200 01 0 00 006041' MOVE S1,[PS.FRC![EXP <SIXBIT/TTY/>,<TIOFFS,,PS.RID!PS.RIA>,<1,,0>]]
1368 001640' 047 01 0 00 000136 PISYS. S1, ;Disable terminal char avail ints
1369 001641' 260 17 0 00 001635* $TEXT (,<? Can't remove TTY interrupts - error code ^0/S1/>)
1370 001643' 200 01 0 00 010762' MOVE S1,[PS.FRC![EXP .PCSTP,<CCOFFS,,0>,<1,,0>]]
1371 001644' 047 01 0 00 000136 PISYS. S1, ;Disable control-C ints
1372 001645' 260 17 0 00 001641* $TEXT (,<? Can't remove control-C interrupts - error code ^0/S1/>)
1373 001647' 263 17 0 00 000000 $RET ;Return
1374
1375 001650' 201 01 0 00 000037 RESTO2: MOVEI S1,.TICTI ;Clear interrupts
1376 001651' 104 00 0 00 000140 DTIX ;De-assign all type-in
1377 001652' 201 01 0 00 000003 MOVEI S1,.TICCC ;Clear interrupt
1378 001653' 104 00 0 00 000140 DTIX ;for control-C trap
1379
1380 001654' 201 01 0 00 400000 MOVEI S1,.FHSLF ;This fork
1381 001655' 200 02 0 00 010527' MOVX S2,ONCHNS ;Channels to inactivate
1382 001656' 104 00 0 00 000133 DIC% ;De-activate all channels
1383 001657' 104 00 0 00 000130 DIR% ;Disable system
1384 001660' 104 00 0 00 000141 CIS% ;Clear interrupt system
1385
1386 001661' 336 00 0 00 005722' SKIPN TTMOD ;Do not 'restore terminal' if we have
1387 001662' 254 00 0 00 001672' JRST RESTX ; not previously done INIT1 above
1388
1389 001663' 201 01 0 00 000100 MOVEI S1,.PRIIN ; Specified terminal
1390 001664' 200 02 0 00 005722' MOVE S2,TTMOD ; Get orig JFN mode word
1391 001665' 104 00 0 00 000110 SFMOD% ; Restore program related modes
1392 001666' 104 00 0 00 000217 STPAR% ; Restore device related modes
1393
1394 001667' 201 01 0 00 400000 MOVEI S1,.FHSLF ;Process handle
1395 001670' 200 02 0 00 005723' MOVE S2,TTIWD ;Get orig terminal interrupt word
1396 001671' 104 00 0 00 000174 STIW% ; and restore orig term interr word
1397
1398 001672' 402 00 0 00 005722' RESTX: SETZM TTMOD ;Clear for next time
1399 001673' 263 17 0 00 000000 $RET

```

```

1400
1401      ;* CLREFL - Clear event flag table
1402
1403      001674' 304 00 0 00 000000      CLREFL: $SAVE    <S1>
1404      001700' 402 00 0 00 005654'      SETZM      EFLST      ;Zero 1'st loc
1405      001701' 200 01 0 00 011003'      MOVE       S1,[XWD EFLST,EFLST+1]
1406      001702' 251 01 0 00 005660'      BLT        S1,EFLN
1407      001703' 263 17 0 00 000000      $RET
1408
1409      ;* BUFPTR -- Reset all Buffer Pointers
1410
1411      001704' 201 01 0 00 003325'      BUFPTR: MOVEI    S1,MBUF2      ;Message Buffers
1412      001705' 202 01 0 00 003251'      MOVEM     S1,MBUF1
1413      001706' 201 01 0 00 003401'      MOVEI     S1,MBUF3
1414      001707' 202 01 0 00 003325'      MOVEM     S1,MBUF2
1415      001710' 201 01 0 00 003455'      MOVEI     S1,MBUF4
1416      001711' 202 01 0 00 003401'      MOVEM     S1,MBUF3
1417      001712' 201 01 0 00 003531'      MOVEI     S1,MBUF5
1418      001713' 202 01 0 00 003455'      MOVEM     S1,MBUF4
1419      001714' 201 01 0 00 003605'      MOVEI     S1,MBUF6
1420      001715' 202 01 0 00 003531'      MOVEM     S1,MBUF5
1421      001716' 201 01 0 00 003661'      MOVEI     S1,MBUF7
1422      001717' 202 01 0 00 003605'      MOVEM     S1,MBUF6
1423      001720' 201 01 0 00 003735'      MOVEI     S1,MBUF8
1424      001721' 202 01 0 00 003661'      MOVEM     S1,MBUF7
1425      001722' 201 01 0 00 004011'      MOVEI     S1,MBUF9
1426      001723' 202 01 0 00 003735'      MOVEM     S1,MBUF8
1427      001724' 201 01 0 00 004065'      MOVEI     S1,MBUF10
1428      001725' 202 01 0 00 004011'      MOVEM     S1,MBUF9
1429      001726' 201 01 0 00 004141'      MOVEI     S1,MBUF11
1430      001727' 202 01 0 00 004065'      MOVEM     S1,MBUF10
1431      001730' 201 01 0 00 004215'      MOVEI     S1,MBUF12
1432      001731' 202 01 0 00 004141'      MOVEM     S1,MBUF11
1433      001732' 402 01 0 00 004215'      SETZM     S1,MBUF12      ;end
1434
1435      001733' 201 01 0 00 004520'      MOVEI     S1,DBUF2      ;Datagram Buffers
1436      001734' 202 01 0 00 004271'      MOVEM     S1,DBUF1
1437      001735' 201 01 0 00 004747'      MOVEI     S1,DBUF3
1438      001736' 202 01 0 00 004520'      MOVEM     S1,DBUF2
1439      001737' 201 01 0 00 005176'      MOVEI     S1,DBUF4
1440      001740' 202 01 0 00 004747'      MOVEM     S1,DBUF3
1441      001741' 201 01 0 00 005425'      MOVEI     S1,DBUF5
1442      001742' 202 01 0 00 005176'      MOVEM     S1,DBUF4
1443      001743' 402 01 0 00 005425'      SETZM     S1,DBUF5
1444
1445      001744' 263 17 0 00 000000      $RET

```

```

1446          SUBTTL Show Network Command
1447
1448          ;* SNET - "SHOW NETWORK" command.
1449
1450          ; Show an abbreviated network configuration data returned in the
1451          ; .SSRCD arg block in the following format:
1452
1453          :
1454          :      Node      Type
1455          :      1.       T-10
1456          :      15.      HSC
1457
1458          SNET:  $SAVE    <T2>
1459                 $TEXT    (<,<      Node      Type>)
1460
1461                 SETOM    EXPINV          ;'Node not present' errors expected
1462                 SETZ     T2,             ;Node#
1463
1464          SNET1:  $CALL    SSRCD           ;Read configuration data
1465                 SKIPA                    ; node not there, try another
1466                 $CALL    PRRCD1         ;Print abbreviated returned data
1467                 AOS      T2             ;Increment to next node
1468                 CAIG     T2,17          ;All nodes attempted?
1469                 JRST     SNET1          ; no, do another
1470                 SETZM    EXPINV         ;Clear the no report flag
1471                 $RET
1472
1473          ; PRRCD1 - Print only node # and dest software type ret from .SSRCD arg block
1474
1475          PRRCD1: MOVE     S1,.$QDST+RCDARG ;Load dest software type
1476                 $CALL    P4A8           ;Convert to 7 bit ASCII2 in S2
1477                 $TEXT    (<,<      ^D3R /.SQOSB+RCDARG/.  ^T/S2/>)
1478                 $RET
1479
1480          ;This subroutine makes 4 left justified ASCII 8 bit bytes into one word ASCII2.
1481          ;To use this routine, put the word in S1, the result will be returned in S2.
1482
1483          P4A8:  $SAVE    <T1,T2,T3,T4>   ;Save scratch ACs
1484                 MOVE     T1,[POINT 8,S1] ;Load the pointer to the input data
1485                 MOVE     T2,[POINT 7,S2] ;Load the pointer to the output data
1486                 SETZ     S2,             ;Make ASCII2
1487                 MOVEI    T4,4            ;Load byte count
1488
1489          P4A81:  ILDB     T3,T1           ;Get a byte
1490                 IDPB     T3,T2          ;Store a byte
1491                 SOJG     T4,P4A81       ;Loop for them
1492                 $RET                    ;Return
  
```

```

1492          SUBTTL Show Node Command
1493
1494          ; * SNOD - Entered by the "SHOW NODE n" command
1495          ;
1496          ; Show the entire configuration data returned in the .SSRCD arg block.
1497          ; Call with NODNUM set to the node desired. Output in the following
1498          ; format:
1499          ;
1500          ; Node 15.
1501          ; Virtual circuit state: 3, Open
1502          ; Maximum datagram size: 264.
1503          ; Maximum message size: 992.
1504          ; Software type: HSC
1505          ; Software version: Y25G
1506          ; Hardware type: HS50
1507          ; Node name: HSC015
1508          ;
1509          ; Port characteristic word: 400000000100
1510          002003' 304 00 0 00 000000
1511          002007' 200 04 0 00 005702'
1512          002010' 260 17 0 00 002312'
1513          002011' 254 00 0 00 002013'
1514          002012' 260 17 0 00 002014'
1515          002013' 263 17 0 00 000000
1516
          SNOD:  $SAVE  <T2>
                  MOVE   T2,NODNUM
                  $CALL  SSRCD
                  JRST   SNODX
                  $CALL  PRRCD2
          SNODX: $RET

```

```

;Node number
;Read config data
; error, exit
;Print all returned data

```

```

1517
1518 ; PRRCD2 - Print all the Returned Configuraton Data from .SSRCD arg block
1519
1520 002014' 304 00 0 00 000000 PRRCD2: $SAVE <S1,S2,T1,T2,T3,T4>
1521
1522 002025' 200 01 0 00 002355' MOVE S1,.SQDSV+RCDARG ;Load dest software ver
1523 002026' 260 17 0 00 001772' $CALL P4A8 ;Convert to 4 ASCII 7 bit bytes
1524 002027' 200 03 0 00 000002 MOVE T1,S2 ;Save in T1
1525
1526 002030' 200 01 0 00 002360' MOVE S1,.SQDHT+RCDARG ;Load dest hardware type
1527 002031' 260 17 0 00 001772' $CALL P4A8 ;Convert to 4 ASCII 7 bit bytes
1528 002032' 200 04 0 00 000002 MOVE T2,S2 ;Save that in T2
1529
1530 002033' 200 01 0 00 002364' MOVE S1,.SQNNM+RCDARG ;Get first node name word
1531 002034' 260 17 0 00 001772' $CALL P4A8 ;Convert to 4 ASCII 7 bit bytes
1532 002035' 200 05 0 00 000002 MOVE T3,S2 ;Save in T3
1533 002036' 200 01 0 00 002365' MOVE S1,.SQNNM+1+RCDARG ;Get second node name word
1534 002037' 260 17 0 00 001772' $CALL P4A8 ;Convert to 4 ASCII 7 bit bytes
1535 002040' 200 06 0 00 000002 MOVE T4,S2 ;Save in T4
1536
1537 002041' 200 01 0 00 002354' MOVE S1,.SQDST+RCDARG ;Load remote software type
1538 002042' 260 17 0 00 001772' $CALL P4A8 ;Convert to 4 ASCII 7 bit bytes in S2
1539
1540 002043' 554 01 0 00 002347' HLRZ S1,.SQVCS+RCDARG ;Load vc status
1541 002044' 331 00 0 00 000001 SKIPL S1 ;Less than 0?
1542 002045' 303 01 0 00 000003 CAILE S1,VC.OPN ;no, .gt. max?
1543 002046' 201 01 0 00 000004 MOVEI S1,VC.OPN+1 ;Yes, force "illegal" print
1544 MOVE S1,[ EXP [ASCIZ/Closed/],[ASCIZ/Start sent/]
1545 ; EXP [ASCIZ/Start received/],[ASCIZ/Open/]
1546 002047' 200 01 0 01 011050' ; EXP [ASCIZ/Illegal/]](S1)
1547
1548 $TEXT (,<Node ^D/.SQOSB+RCDARG/.
1549 Virtual circuit state: ^D/.SQVCS+RCDARG,LHMASK/, ^T/(S1)/
1550 Maximum datagram size: ^D/.SQMDD+RCDARG/.
1551 Maximum message size: ^D/.SQMDM+RCDARG/.
1552 Software type: ^T/S2/
1553 Software version: ^T/T1/
1554 Hardware type: ^T/T2/
1555 Node name: ^T/T3/^T/T4/
1556 002050' 260 17 0 00 001767* Port characteristic word: ^D/.SQPCW+RCDARG/>)
1557 002052' 263 17 0 00 000000 $RET

```

```

1558          SUBTTL  DUP Interface Routines
1559
1560          ;* DUSTST - Get status of DUP Server at remote node.
1561          ;
1562          ; Interrogate the DUP Server to determine its characteristics,
1563          ; its state and state of program currently running (if any).
1564          ; It is legal in IDLE and ACTIVE states and does not affect the server state
1565          ; It has a timeout interval of 3 seconds.
1566          ; If response times out, the host should break the connection.
1567          ;
1568          ; Enter the SSSMG subroutine with:      T2 = Cmd Msg buffer address
1569          ;                                       T3 = Cmd Msg buffer length in bytes
1570          ; Ret+1 if jsys failure
1571          ; Ret+2 if success.
1572
1573          002053' 304 00 0 00 000000          DUSTST: $SAVF  <T2,T3>          ;Save some ac's
1574          002060' 350 04 0 00 005711'        AOS      T2,CMDREF
1575          002061' 242 04 0 00 000004          LSH      T2,4          ;Shift to byte 3
1576          002062' 202 04 0 00 002075'        MOVEM   T2,STSBLK          ; and put in command msg
1577
1578          002063' 201 04 0 00 000001          MOVEI   T2,OPCGDS          ;Opcode 1
1579          002064' 202 04 0 00 005724'        MOVEM   T2,CUROPC          ;Save in Current Opcode reg
1580          002065' 241 04 0 00 777770          ROT      T2,-^D8          ;Shift to byte 0
1581          002066' 202 04 0 00 002077'        MOVEM   T2,STSBLK+2        ;Put in arg block
1582
1583          002067' 201 04 0 00 002075'        MOVEI   T2,STSBLK          ;Cmd msg buffer address
1584          002070' 201 05 0 00 000014'        MOVEI   T3,STSLN*4        ;Cmd msg buffer length in bytes
1585          002071' 260 17 0 00 002530'        $CALL   SSSMG          ;Send 'GET DUST STATUS'
1586          002072' 334 00 0 00 000000          SKIPA          ; error
1587          002073' 350 00 0 17 000000          AOS      (P)          ;Ret+2
1588          002074' 203 17 0 00 000000          $RET
1589
1590          ;DUST STATUS COMMAND MESSAGE BUFFER
1591
1592          002075' 000000 000000          STSBLK: 0          ;Cmd ref #
1593          002076' 000000 000000          0          ;not used
1594          002077' 000000 000000          0          ;Opcode,rsvd,Modifiers
1595          000003          STSLN=-STSBLK          ;Length
1596

```

```

1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612 002103' 304 00 0 00 000000
1613 002105' 350 04 0 00 005711'
1614 002106' 242 04 0 00 000004'
1615 002107' 202 04 0 00 002122'
1616
1617 002110' 201 04 0 00 000003'
1618 002111' 202 04 0 00 005724'
1619 002112' 241 04 0 00 777770'
1620 002113' 202 04 0 00 002124'
1621
1622 002114' 201 04 0 00 002122'
1623 002115' 201 05 0 00 000022'
1624 002116' 260 17 0 00 002530'
1625 002117' 334 00 0 00 000000'
1626 002120' 350 00 0 17 000000'
1627 002121' 202 17 0 00 000000'
1628
1629
1630
1631 002122' 000000 000000
1632 002123' 000000 000000
1633 002124' 000000 000000
1634 002125' 120 123 105 125 00
1635 002126' 104 117 0000000
1636
1637

; * EXLPGM - Send DUP 'EXECUTE LOCAL PROGRAM' request
;
; Receipt of this command causes the controller to search its local media for
; the named program, load it and execute it.
; Receipt of a successful response by the host means that the program is
; executing and the server is in the ACTIVE state.
; This command is only legal when the server is in the IDLE state.
; The timeout value for this command is specified in the GET DUST STATUS resp.
;
; Enter the SSSMG subroutine with:      T2 = Cmd Msg buffer address
;                                       T3 = Cmd Msg buffer length in bytes
; Ret+1 if jsys failure
; Ret+2 if successful

EXLPGM: $SAVE      <T2,T3>              ;Save some ac's
        AOS        T2,CMDREF
        LSH        T2,4
        MOVEM      T2,EXLBLK            ;Shift to byte 3
                                           ; and put in cmd msg
        MOVEI      T2,OPCELP            ;Opcode 3
        MOVEM      T2,CUROPC            ;Save in Current Opcode reg
        ROT        T2,-^D8              ;Shift to byte 0
        MOVEM      T2,EXLBLK+2          ;Put in arg block
        MOVEI      T2,EXLBLK            ;Cmd msg buffer address
        MOVEI      T3,EXLLEN*4-2        ;Cmd msg buffer length in bytes
        $CALL      SSSMG                ;Send 'EXECUTE LOCAL PROGRAM'
        SKIPA      ; error
        AOS        (P)                  ;Ret+2
        $RET

;EXECUTE LOCAL PROGRAM COMMAND MSG BUFF

EXLBLK: 0                                ;Cmd ref#
        0                                ;not used
        0                                ;Opcode,rsvd,Modifiers
        BYTE (8) 'P',' ','S',' ','E',' ','U' ;Pcm name in ascii (pseudo)
        BYTE (8) 'D',' ','O',' ',' ',' ',' ' ;' ', n/u
        EXLLEN=.-EXLBLK                ;Length

```

```

1638
1639 ; SEND - Send DUP "SEND DATA" Command
1640 ;
1641 ; This command is used to pass info between the initiating host program and
1642 ; the remote program, otherwise they may be optionally implemented.
1643 ; This command is legal only when the server is in the ACTIVE state.
1644 ; If the remote program terminates abnormally, putting the server back in the
1645 ; IDLE state, outstanding SEND/RCV DATA commands may be lost.
1646 ; In the event that the specified timeout interval is exceeded, the host
1647 ; should issue a GET DUST STATUS command to see if the remote program is still
1648 ; running (DUP server active). If it is, the progress indicator should be
1649 ; remembered and the timeout interval reinstated. If the second timeout
1650 ; expires and a second DUST STATUS shows remote program made no progress, the
1651 ; program should be considered broken and should be aborted.
1652 ; Enter the SSSMG subroutine with:      T2 = Cmd Msg buffer address
1653 ;                                       T3 = Cmd Msg buffer length in bytes
1654 ;
1655 ; Ret+1 if jsys failure
1656 ; Ret+2 if successful
1657 002127' 304 00 0 00 000000 SEND: $SAVE <T2,T3> ;Save some ac's
1658 002134' 350 04 0 00 005711' AOS T2,CMDREF
1659 002135' 242 04 0 00 000004 LSH T2,4 ;Shift to byte 3
1660 002136' 202 04 0 00 002151' MOVEM T2,SNDBLK ; and put in cmd msg
1661
1662 002137' 201 04 0 00 000004 MOVEI T2,OPCSD ;Opcode 4
1663 002140' 202 04 0 00 005724' MOVEM T2,CUROPC ;Save in Current Opcode reg
1664 002141' 241 04 0 00 777770 ROT T2,-^D8 ;Shift to byte 0
1665 002142' 202 04 0 00 002153' MOVEM T2,SNDBLK+2 ;Put in arg block
1666
1667 002143' 201 04 0 00 002151' MOVEI T2,SNDBLK ;Cmd msg buffer address
1668 002144' 201 05 0 00 000034' MOVEI T3,SNDBLK*4 ;Cmd msg buffer length in bytes
1669 002145' 260 17 0 00 002530' $CALL SSSMG ;Send "SEND DATA"
1670 002146' 334 00 0 00 000000 $CALL SKIPA ; error
1671 002147' 350 00 0 17 000000 AOS (P) ;ret+2
1672 002150' 263 17 0 00 000000 $RET
1673
1674 ;SEND DATA COMMAND MSG BUFFER
1675
1676 002151' 000000 000000 SNDBLK: 0 ;Cmd ref#
1677 002152' 000000 000000 0 ;not used
1678 002153' 000000 000000 0 ;Opcode,rsvd,Modifiers
1679 002154' 000000 000000 0 ;Byte count
1680 002155' 000000 000000 0 ;Buffer descriptor
1681 002156' 000000 000000 0 ;
1682 002157' 000000 000000 0 ;
1683 000007 SNDLEN=-SNDBLK ;Length

```

```

1684
1685      ; SENDIC - Send DUP 'SEND DATA IMMEDIATE' Command
1686
1687      ; This command is used to pass info between the initiating host program and
1688      ; the remote program, otherwise they may be optionally implemented.
1689      ; This command is legal only when the server is in the ACTIVE state.
1690      ; If the remote program terminates abnormally, putting the server back in the
1691      ; IDLE state, outstanding SEND/RCV DATA commands may be lost.
1692      ; In the event that the specified timeout interval is exceeded, the host
1693      ; should issue a GET DUST STATUS command to see if the remote program is still
1694      ; running (DUP server active). If it is, the progress indicator should be
1695      ; remembered and the timeout interval reinstated. If the second timeout
1696      ; expires and a second DUST STATUS shows remote program made no progress, the
1697      ; program should be considered broken and should be aborted.
1698      ; Enter the SSSMG subroutine with:      T2 = Cmd Msg buffer address
1699      ;                                         T3 = Cmd Msg buffer length in bytes
1700      ; Ret+1 if jsys failure
1701      ; Ret+2 if successful
1702
1703      002160' 304 00 0 00 000000      SENDIC: $SAVE      <T2,T3>      ;Save some ac's
1704      002165' 350 04 0 00 005711'      AOS      T2,CMDREF
1705      002166' 242 04 0 00 000004'      LSH      T2,4      ;Shift to byte 3
1706      002167' 202 04 0 00 002202'      MOVEM     T2,SDCBLK      ; and put in cmd msg
1707
1708      002170' 201 04 0 00 000007'      MOVEI     T2,OPCSDI      ;Opcode 7
1709      002171' 202 04 0 00 005724'      MOVEM     T2,CUROPC      ;Save in Current Opcode reg
1710      002172' 241 04 0 00 777770'      ROT      T2,-^D8      ;Shift to byte 0
1711      002173' 202 04 0 00 002204'      MOVEM     T2,SDCBLK+2      ;Put in arg block
1712
1713      002174' 201 04 0 00 002202'      MOVEI     T2,SDCBLK      ;Cmd msg buffer address
1714      002175' 201 05 0 00 000044'      MOVEI     T3,SDCLEN*4      ;Cmd msg buffer length in bytes
1715      002176' 260 17 0 00 002530'      $CALL     SSSMG      ;Send 'SEND DATA IMMEDIATE'
1716      002177' 334 00 0 00 000000'      SKIPA      ; error
1717      002200' 350 00 0 17 000000'      AOS      (P)      ;ret+2
1718      002201' 263 17 0 00 000000'      $RET
1719
1720      ;SEND DATA COMMAND MSG BUFFER
1721
1722      002202' 000000 000000      SDCBLK: 0      ;Cmd ref#
1723      002203' 000000 000000      0      ;not used
1724      002204' 000000 000000      0      ;Opcode, rsvd, Modifiers
1725      002205' 000000 000000      0      ;2 byte count, 2 msg data bytes
1726      002206'      BLOCK TTISIZ/4      ;Message data
1727      000011      SDCLEN=-SDCBLK      ;Length
  
```

```

1728
1729      ; SENDIR - Send DUP "SEND DATA IMMEDIATE" Response
1730      ;
1731      ; This is the Response passed to the HSC after receiving a Send Data Immed
1732      ; from the HSC.
1733      ;
1734      ; Enter the SSSMG subroutine with:      T2 = Cmd Msg buffer address
1735      ;                                       T3 = Cmd Msg buffer length in bytes
1736      ; Ret+1 if jsys failure
1737      ; Ret+2 if successful
1738
1739 002213' 304 00 0 00 000000 SENDIR: $SAVE <T2,T3>      ;Save some ac's
1740 002220' 200 04 0 00 005712' MOVE T2,SDIREF      ;Get cmd ref# saved from SDI from HSC
1741 002221' 202 04 0 00 002233' MOVEM T2,SDRBLK      ; and make it response ref # in arg
1742
1743 002222' 201 04 0 00 000207 MOVEI T2,OPCSDI+OPCRES      ;Opcode 207
1744 002223' 241 04 0 00 777770 ROT T2,-^D8      ; Shift to byte 0
1745 002224' 202 04 0 00 002235' MOVEM T2,SDRBLK+2      ; Put in arg block
1746
1747 002225' 201 04 0 00 002233' MOVEI T2,SDRBLK      ;Response msg buffer address
1748 002226' 201 05 0 00 000014' MOVEI T3,SDRLEN*4      ;Response msg buffer length in bytes
1749 002227' 260 17 0 00 002530' $CALL SSSMG      ;Send "SEND DATA IMMED" response
1750 002230' 334 00 0 00 000000 $SKIP      ; error
1751 002231' 350 00 0 17 000000 AOS (P)      ;ret+2
1752 002232' 263 17 0 00 000000 $RET
1753
1754      ; SEND DATA COMMAND MSG BUFFER
1755
1756 002233' 000000 000000 SDRBLK: 0      ;Response ref#
1757 002234' 000000 000000 0      ;not used
1758 002235' 000000 000000 0      ;Opcode,rsvd,Modifiers
1759      SDRLEN=-SDRBLK      ;Length
  
```

```

1760
1761      ; ABORT - Send the DUP "ABORT PROGRAM" request.
1762      ;
1763      ; This command is used to terminate the execution of a remote program in
1764      ; an orderly fashion. When a successful response is received to this command
1765      ; the remote program has stopped executing and the server is in the IDLE state.
1766      ; Note that the sending of this command does not preclude further SEND/RCV
1767      ; DATA exchanges. On the contrary, the remote program may be designed to send
1768      ; out termination status and possibly ask questions during its forced exit.
1769      ; The timeout for this command is a fixed 10 seconds, and if a response is not
1770      ; received by then, the connection to the DUST should be terminated.
1771      ; This command is legal only if the DUST is in the ACTIVE state.
1772      ;
1773      ; Enter the SSSMG subroutine with:      T2 = Cmd Msg buffer address
1774      ;                                       T3 = Cmd Msg buffer length in bytes
1775      ; Ret+1 if jsys failure
1776      ; Ret+2 if successful
1777
1778 002236' 304 00 0 00 000000      ABORT: $SAVE <T2,T3>      ;Save some ac's
1779 002243' 350 04 0 00 005711'      AOS T2,CMDREF
1780 002244' 242 04 0 00 000004      LSH T2,4      ;Shift to byte 3
1781 002245' 202 04 0 00 002260'      MOVEM T2,ABTBLK      ; and put in cmd msg
1782
1783 002246' 201 04 0 00 000006      MOVEI T2,OPCABT      ;Opcode 6
1784 002247' 202 04 0 00 005724'      MOVEM T2,CUOPC      ;Save in Current Opcode reg
1785 002250' 241 04 0 00 777770      ROT T2,-^D8      ;Shift to byte 0
1786 002251' 202 04 0 00 002262'      MOVEM T2,ABTBLK+2      ;Put in arg block
1787
1788 002252' 201 04 0 00 002260'      MOVEI T2,ABTBLK      ;Cmd msg buffer address
1789 002253' 201 05 0 00 000014'      MOVEI T3,ABTLEN*4      ;Cmd msg buffer length in bytes
1790 002254' 263 17 0 00 002530'      $CALL SSSMG      ;Send "RECEIVE DATA"
1791 002255' 334 00 0 00 000000      SKIPA      ; error
1792 002256' 350 00 0 17 000000      AOS (P)      ;Ret+2
1793 002257' 263 17 0 00 000000      $RET
1794
1795      ;ABORT COMMAND MESSAGE BUFFER
1796
1797 002260' 000000 000000      ABTBLK: 0      ;Cmd ref#
1798 002261' 000000 000000      0      ;not used
1799 002262' 000000 000000      0      ;Opcode,rsvd,Modifiers
1800      ABTLEN=-ABTBLK      ;Length
  
```

```

1801                      SUBTTL  SCS JSYS/UUO Execution
1802
1803                      ;* SCAINT - Add (set up) Interrupt Channels for SCA related events
1804                      ;
1805                      ;       Ret+1   if jsys failure
1806                      ;       Ret+2   if successful
1807
1808 002263' 336 00 0 00 001621*  SCAINT: SKIPN  MONTYP           ;TOPS-20?
1809 002264' 254 00 0 00 002303' JRST   SCAINX           ;No, TOPS-10, return now
1810 002265' 304 00 0 00 000000  $SAVE  <S1,S2>         ;Save the ac's
1811 002272' 201 02 0 00 002305' MOVEI  S2,AICARG        ;Arg block addr
1812 002273' 201 01 0 00 000005  MOVEI  S1,AICLEN        ;Arg block length
1813 002274' 202 01 0 02 000000  MOVEM  S1,..SQLEN(S2)      ;Put in arg block
1814
1815 002275' 201 01 0 00 000020  MOVEI  S1,..SSAIC        ;Function code
1816 002276' 260 17 0 00 002751' $CALL  $SCS           ;Do the JSYS
1817 002277' 254 00 0 00 003014' JRST   FLAGIT          ;Flag the error and return +1
1818 002300' 332 00 0 00 005705' SKIPE  TRACE           ;Trace enabled?
1819 002301' 260 17 0 00 002050* $TEXT  (<..SSAIC successful>) ;Yes
1820 002303' 350 00 0 17 000000  SCAINX: AOS   (P)         ;Ret+2
1821 002304' 263 17 0 00 000000  $RET
1822
1823                      ; ARGUMENT BLOCK
1824
1825 002305' 000000 000005  AICARG: 0,AICLEN           ;Ret Words processed,..block length
1826 002306' 000002 000001  .SIDMA,..DMACHN          ;DMA interrupt on ch 1
1827 002307' 000003 000002  .SIPAN,..EVTCHN          ;EVENT interrupt on ch 2
1828 002310' 000001 000003  .SIMSA,..MSGCHN          ;MESSAGE interrupt on ch 3
1829 002311' 000000 000004  .SIDGA,..DGMCHN          ;DATAGRAM interrupt on ch 4
1830                      AICLEN=.-AICARG              ;Length arg block

```

```

1831
1832      ; * SSRCD - Return Connection Data (about a remote node)
1833      ;
1834      ; Enter with T2 = Node# or 0 (Specify CID or Node but NOT both)
1835      ; Return+1 if jsys error
1836      ; Return+2 if successful
1837
1838 002312' 304 00 0 00 000000 SSRCD: $SAVE <S1,S2,T2> ;Save some acs
1839 002320' 201 02 0 00 002344' MOVEI S2,RCDARG ;Arg block addr
1840
1841 002321' 201 01 0 00 000024 MOVEI S1,.LBRCD ;Arg block length
1842 002322' 202 01 0 02 000000 MOVEM S1,.SQLEN(S2) ;Put in arg block
1843 002323' 400 01 0 00 000000 SETZ S1
1844 002324' 202 01 0 02 000001 MOVEM S1,.SQCID(S2) ; " CID = 0
1845 002325' 202 04 0 02 000002 MOVEM T2,.SQOSB(S2) ; " Node#
1846
1847 002326' 201 01 0 00 000011 MOVEI S1,.SSRCD ;Function code
1848 002327' 260 17 0 00 002751' $CALL $SCS ;Do the JSYS
1849 002330' 254 00 0 00 002336' JRST SSRCD1 ; Error ret
1850 002331' 350 00 0 17 000000 AOS (P) ;Ret+2
1851 002332' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
1852 002333' 260 17 0 00 002301* $TEXT (<,.SSRCD successful>) ;Yes
1853 002335' 263 17 0 00 000000 $RET ;Return+2
1854
1855 002336' 200 02 0 00 005727' SSRCD1: MOVE S2,SCSERC ;Load last error code
1856 002337' 302 02 0 00 602621' CAIE S2,SCSISB ;Invalid node number?
1857 002340' 254 00 0 00 003014' JRST FLAGIT ;Flag the error and return +1
1858 002341' 336 00 0 00 005710' SKIPN EXPINV ;Yes, do we expect it?
1859 002342' 254 00 0 00 003014' JRST FLAGIT ;Flag the error and return +1
1860 002343' 263 17 0 00 000000 $RET ;Return +1
1861
1862      ; ARGUMENT BLOCK
1863
1864 002344' 000000 000000 RCDARG: 0 ;Ret #wds processed,,block length
1865 002345' 000000 000000 0 ;CID (specify CID or Node
1866 002346' 000000 000000 0 ;Node # but not both)
1867 002347' 000000 000000 0 ;Ret Virtual Ckt State,,port #
1868 002350' 000000 000000 0 ;Ret Remote sys addr, 8 bytes
1869 002351' 000000 000000 0 ;Ret
1870 002352' 000000 000000 0 ;Ret max DG size,,at dest
1871 002353' 000000 000000 0 ;Ret max MSG
1872 002354' 000000 000000 0 ;Ret softw type at dest, 4 ascii bytes
1873 002355' 000000 000000 0 ;Ret " ver " 4
1874 002356' 000000 000000 0 ;Ret " edit lev " 8
1875 002357' 000000 000000 0 ;Ret
1876 002360' 000000 000000 0 ;Ret hardw type code at dest 4 ascii by
1877 002361' 000000 000000 0 ;Ret " ver at dest, 12 ascii bytes
1878 002362' 000000 000000 0 ;Ret
1879 002363' 000000 000000 0 ;Ret
1880 002364' 000000 000000 0 ;Ret dest port name, 8 ascii bytes
1881 002365' 000000 000000 0 ;Ret
1882 002366' 000000 000000 0 ;Ret local port# (RH20 chan# of CI-20)
1883 002367' 000000 000000 0 ;Ret Port characteristics word

```

```

1884
1885      ; * SSCON - Request a connection with another node on the CI.
1886      ;
1887      ; SCS will return as soon as the connection request has been sent.
1888      ; The calling process will be notified by an EVENT interrupt when the
1889      ; request is processed.
1890      ;
1891      ;       Ret+1   if jsys error or conection rejected
1892      ;       Ret+2   if connection successful
1893      ;
1894      SSCON:  $SAVE    <S1,S2>           ;Save some acs
1895              MOVEI    S2,CONARG         ;Arg block addr
1896
1897              MOVEI    S1,..LBCON        ;Length of argument block
1898              MOVEM    S1,..SQLEN(S2)    ;Put in arg block
1899              MOVE     S1,[POINT 7,SPN]   ;Ptr to source process name
1900              MOVEM    S1,..SQSPN(S2)
1901              MOVE     S1,[POINT 7,DPN]   ;Ptr to destination process name
1902              MOVEM    S1,..SQDPN(S2)
1903              MOVE     S1,..SQDNUM       ;Node#
1904              HRLZM    S1,..SQSYS(S2)    ;Address of connect data block
1905              MOVEI    S1,CONDAT
1906              MOVEM    S1,..SQCDT(S2)
1907              MOVEI    S1,MBUF1          ;Queue addr of first msg buffer chain
1908              MOVEM    S1,..SQAMC(S2)
1909              MOVEI    S1,DBUF1          ;Queue addr of first DG buffer chain
1910              MOVEM    S1,..SQADC(S2)
1911
1912              MOVEI    S1,..SSCON        ;Function code
1913              $CALL    $SCS              ;Do the JSYS
1914              JRST     FLAGIT            ;Flag the error and return +1
1915
1916              MOVE     S1,CONARG+..SQRCI  ;Get returned CID
1917              MOVEM    S1,CID            ; and store it
1918              SKIPE    TRACE             ;Trace enabled?
1919              $TEXT    (<..SSCON successful>) ;Yes
1920              AOS      (P)               ;Skip the return
1921              $RET
1922
1894 002370' 304 00 0 00 000000
1895 002375' 201 02 0 00 002426'
1896
1897 002376' 201 01 0 00 000010
1898 002377' 202 01 0 02 000000
1899 002400' 200 01 0 00 011221'
1900 002401' 202 01 0 02 000001
1901 002402' 200 01 0 00 011222'
1902 002403' 202 01 0 02 000002
1903 002404' 200 01 0 00 005702'
1904 002405' 516 01 0 02 000003
1905 002406' 201 01 0 00 002436'
1906 002407' 202 01 0 02 000004
1907 002410' 201 01 0 00 003251'
1908 002411' 202 01 0 02 000005
1909 002412' 201 01 0 00 004271'
1910 002413' 202 01 0 02 000006
1911
1912 002414' 201 01 0 00 000000
1913 002415' 260 17 0 00 002751'
1914 002416' 25: 00 0 00 003014'
1915
1916 002417' 200 01 0 00 002435'
1917 002420' 202 01 0 00 005701'
1918 002421' 332 00 0 00 005705'
1919 002422' 260 17 0 00 002333*
1920 002424' 350 00 0 17 000000
1921 002425' 263 17 0 00 000000
1922

```

```

1923
1924      ; ARGUMENT BLOCK
1925
1926 002426' 000000 000000      CONARG: 0      ;Ret #words processed,,block length
1927 002427' 000000 000000      0      ;Byte ptr, to ASCII SRC process name
1928 002430' 000000 000000      0      ;DST
1929 002431' 000000 000000      0      ;Node# of dest,,CID
1930 002432' 000000 000000      0      ;Addr of connection data
1931 002433' 000000 000000      0      ;Addr of first buff of MSG buff chain
1932 002434' 000000 000000      0      ;DG
1933 002435' 000000 000000      0      ;Ret CID (32 bits left justified)
1934
1935 002436'      CONDAT: BLOCK 4
1936
1937 002442' 115 131 044 120 122      SPN:  ASCIIZ  /MY$PROGRAM/      ;Src process name
1938 002445' 115 123 103 120 044      DPN:  ASCIIZ  /MSCP$DUP/      ;Dst process name
  
```

```

1939
1940      ; * SSEVT - Get Entry from Event Queue
1941      ;
1942      ; Return+1      if jsys error
1943      ; Return+2      if successful...event code/data in EVTARG+.SQEVT/.SQDTA
1944
1945 002447' 304 00 0 00 000000 SSEVT: $SAVE <S1,S2> ;Save some ac's
1946 002454' 201 02 0 00 002471' MOVEI S2,EVTARG ;Arg block addr
1947
1948 002455' 201 01 0 00 000010 MOVEI S1,.LBEVT ;Arg block length
1949 002456' 202 01 0 02 000000 MOVEM S1,.SQLEN(S2) ;Put in arg block
1950 002457' 200 01 0 00 005701' MOVE S1,CID ;CID or -1
1951 002460' 202 01 0 02 000001 MOVEM S1,.SQCID(S2)
1952
1953 002461' 201 01 0 00 000025 MOVEI S1,.SSEVT ;Function code
1954 002462' 260 17 0 00 002751' $CALL $SCS ;Do the JSYS
1955 002463' 254 00 0 00 002470' JRST SSEVTX ; Error ret (evaluated in EVTSRV)
1956 002464' 350 00 0 17 000000 AOS (P) ;Ret+2
1957 002465' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
1958 002466' 260 17 0 00 002422* $TEXT (<,.SSEVT successful>) ;Yes
1959
1960 002470' 263 17 0 00 000000 SSEVTX: $RET
1961
1962 ; ARGUMENT BLOCK
1963
1964 002471' 000000 000000 EVTARG: 0 ;Ret # wds processed..block length
1965 002472' 000000 000000 0 ;CID or -1, CID on return
1966 002473' 000000 000000 0 ;Reserved..node#
1967 002474' 000000 000000 0 ;Returned event code
1968 002475' 000000 000000 0 ;Returned event data
1969 002476' 000000 000000 0 ;
1970 002477' 000000 000000 0 ;
1971 002500' 000000 000000 0 ;

```

```

1972
1973      ; * SSDIS - Disconnect from a remote node
1974      ;
1975      ; Return+1      if jsys error
1976      ; Return+2      if successful
1977
1978      002501' 304 00 0 00 000000      SSDIS: $SAVE    <S1,S2>      ;Save some ac's
1979      002506' 201 02 0 00 002525'      MOVEI      S2,DISARG      ;Arg block addr
1980
1981      002507' 201 01 0 00 000003      MOVEI      S1,.LBDIS      ;Arg block length
1982      002510' 202 01 0 02 000000      MOVEM      S1,.SQLEN(S2)      ;Put in arg block
1983      002511' 200 01 0 00 005701'      MOVE       S1,CID
1984      002512' 202 01 0 02 000001      MOVEM      S1,.SQCID(S2)      ; " CID
1985      002513' 400 01 0 00 000000      SETZ       S1,
1986      002514' 202 01 0 02 000002      MOVEM      S1,.SQDIS(S2)      ; " reason for disconnect
1987
1988      002515' 201 01 0 00 000003      MOVEI      S1,.SSDIS      ;Function code
1989      002516' 260 17 0 00 002751'      $CALL      $SCS      ;Do the JSYS
1990      002517' 254 00 0 00 003014'      JRST      FLAGIT      ;Flag the error and return +1
1991      002520' 350 00 0 17 000000      AOS        (P)      ;Ret+2
1992      002521' 332 00 0 00 005705'      SKIPE      TRACE      ;Trace enabled?
1993      002522' 260 17 0 00 002466*      $TEXT      (<,.SSDIS successful>) ;Yes
1994      002524' 263 17 0 00 000000      $RET
1995
1996      ; ARGUMENT BLOCK
1997
1998      002525' 000000 000000      DISARG: 0      ;Ret #wds processed,.block length
1999      002526' 000000 000000      0      ;CID
2000      002527' 000000 000000      0      ;Disconnect code, reason for disconn
  
```

```

2001
2002      ; * SSSMG - Send a message to a remote node.
2003      ;
2004      ; Enter with:  T2 = Message buffer Address
2005      ;               T3 = Message buffer length in bytes
2006      ;
2007      ; Return+1      if jsys error
2008      ; Return+2      if successful
2009
2010      002530' 304 00 0 00 000000' SSSMG: $SAVE <S1,S2,T2,T3>
2011      002537' 201 02 0 00 002557'      MOVEI S2,SMGARG      ;Arg block addr
2012
2013      002540' 201 01 0 00 000005'      MOVEI S1,.LBSMG      ;Arg block length
2014      002541' 202 01 0 02 000000'      MOVEM S1,.SQLEN(S2)      ;Put in arg block
2015      002542' 200 01 0 00 005701'      MOVE S1,CID
2016      002543' 202 01 0 02 000001'      MOVEM S1,.SQCID(S2)      ; " CID
2017      002544' 202 04 0 02 000002'      MOVEM T2,.SQAPT(S2)      ; " Addr of msg buffer
2018      002545' 202 05 0 02 000003'      MOVEM T3,.SQLPT(S2)      ; " Length
2019      002546' 402 00 0 02 000004'      SETZM .SQFLG(S2)      ;Use industry Compatible mode
2020
2021      002547' 201 01 0 00 000006'      MOVEI S1,.SSSMG      ;Function code
2022      002550' 260 17 0 00 002751'      $CALL $SCS      ;Do the JSYS
2023      002551' 254 00 0 00 003014'      JRST FLAGIT      ;Flag the error and return +1
2024      002552' 350 00 0 17 000000'      AOS (P)      ;Ret+2
2025      002553' 332 00 0 00 005705'      SKIPE TRACE      ;Trace enabled?
2026      002554' 260 17 0 00 002522*      $TEXT (,<.SSSMG successful>) ;Yes
2027      002556' 263 17 0 00 000000'      $RET
2028
2029      ; ARGUMENT BLOCK
2030
2031      002557' 000000 000000' SMGARG: 0      ;Ret #wds processed,,block length
2032      002560' 000000 000000'      0      ;CID
2033      002561' 000000 000000'      0      ;Addr of msg text
2034      002562' 000000 000000'      0      ;Length of msg 8 bit bytes for ind std
2035      002563' 000000 000000'      0      ;Flags

```

```

2036
2037      ; * SSQRM - Queue Message Buffers - receive a message from a DUP server.
2038      ;
2039      ; Zero the first location of the buffer and re-queue it.
2040      ; There is no Event Interrupt as a result of issuing .SSQRM
2041      ;
2042      ; Return+1      if jsys error
2043      ; Return+2      if successful
2044
2045 002564' 304 00 0 00 000000      SSQRM: $SAVE    <S1,S2>          ;Save some ac's
2046 002571' 201 02 0 00 002611'    MOVEI    S2,QRMARG        ;Arg block addr
2047
2048 002572' 201 01 0 00 000003      MOVEI    S1,.,LBQRM        ;Arg block length
2049 002573' 202 01 0 02 000000      MOVEM    S1,.,SQLEN(S2)    ;Put in arg block
2050 002574' 200 01 0 00 005701'    MOVE     S1,CID          ; " CID
2051 002575' 202 01 0 02 000001      MOVEM    S1,.,SQCID(S2)    ;Addr of msg buffer
2052 002576' 200 01 0 00 005703'    MOVE     S1,BUFADR        ;Clear 1'st loc
2053 002577' 402 00 1 00 000001      SETZM    @S1              ;Put addr of msg buffer to arg block
2054 002600' 202 01 0 02 000002      MOVEM    S1,.,SQAQFB(S2)
2055
2056 002601' 201 01 0 00 000007      MOVEI    S1,.,SSQRM        ;Function code
2057 002602' 260 17 0 00 002751'    $CALL    $SCS            ;Do the JSYS
2058 002603' 254 00 0 00 003014'    JRST     FLAGIT        ;Flag the error and return +1
2059 002604' 350 00 0 17 000000      AOS      (P)            ;Ret+2
2060 002605' 332 00 0 00 005705'    SKIPE    TRACE        ;Trace enabled?
2061 002606' 260 17 0 00 002554*    $TEXT    (,.<.SSQRM successful>) ;Yes
2062 002610' 263 17 0 00 000000      $RET
2063
2064      ; ARGUMENT BLOCK
2065
2066 002611' 000000 000000      QRMARG: 0          ;Ret #wds processed,.,block length
2067 002612' 000000 000000      0              ;CID
2068 002613' 000000 000000      0              ;Addr of 1st msg buff in msg buff chain
  
```

```

2069
2070      ; * SSCSP - Connect State Poll - status of connection
2071      ;
2072      ; Enter with T2 = Byte pointer to dest process name
2073      ; Return+1 if jsys error
2074      ; Return+2 if successful
2075
2076      002614' 304 00 0 00 000000 SSCSP: $SAVE <S1,S2,T2> ;Save some ac's
2077      002622' 201 02 0 00 002640' MOVEI S2,CSPARG ;Arg block addr
2078
2079      002623' 201 01 0 00 000007 MOVEI S1,..LBCSP ;Arg block length
2080      002624' 202 01 0 02 000000 MOVEM S1,..SQLEN(S2) ;Put in arg block
2081      002625' 200 01 0 00 005701' MOVE S1,CID ;
2082      002626' 202 01 0 02 000001 MOVEM S1,..SQCID(S2) ; " CID
2083      002627' 202 04 0 02 000004 MOVEM T2,..SQBDN(S2) ; " Dest name byte pointer
2084
2085      002630' 201 01 0 00 000010 MOVEI S1,..SSCSP ;Function code
2086      002631' 260 17 0 00 002751' $CALL $SCS ;Do the JSYS
2087      002632' 254 00 0 00 003014' JRST FLAGIT ;Flag the error and return +1
2088      002633' 350 00 0 17 000000 AOS (P) ;Ret+2
2089      002634' 332 00 0 00 005705' SKIPE TRACE ;Trace enabled?
2090      002635' 260 17 0 00 002606* $TEXT (,<.SSCSP successful>) ;Yes
2091      002637' 263 17 0 00 000000 $RET
2092
2093      ; ARGUMENT BLOCK
2094
2095      002640' 000000 000000 CSPARG: 0 ;Ret #wds processed,,block length
2096      002641' 000000 000000 0 ;CID
2097      002642' 000000 000000 0 ;Ret connection state
2098      002643' 000000 000000 0 ;Ret dest connect
2099      002644' 000000 000000 0 ;Ret hYTE ptr of loc to start dest name
2100      002645' 000000 000000 0 ;Ret node#
2101      002646' 000000 000000 0 ;Ret src disconn code,,dst disconn code
  
```

```

2102
2103      ; * SSSTS - Status Info on Connection (Fast form of .SSCSP)
2104      ;
2105      ; Return+1      if jsys error
2106      ; Return+2      if successful
2107
2108 002647' 304 00 0 00 000000      SSSTS: $SAVE      <S1,S2>      ;Save some ac's
2109 002654' 201 02 0 00 002671'      MOVEI      S2,STSARG      ;Arg block addr
2110
2111 002655' 201 01 0 00 000004      MOVEI      S1,.LBSTS      ;Arg block length
2112 002656' 202 01 0 02 000000      MOVEM      S1,.SQLEN(S2)      ;Put in arg block
2113 002657' 200 01 0 00 005701'      MOVE      S1,CID
2114 002660' 202 01 0 02 000001      MOVEM      S1,.SQCID(S2)      ; " CID
2115
2116 002661' 201 01 0 00 000012      MOVEI      S1,.SSSTS      ;Function code
2117 002662' 260 17 0 00 002751'      $CALL      $SCS      ;Do the JSYS
2118 002663' 254 00 0 00 002670'      JRST      SSSTSX      ; Error ret (evaluated in DISCON)
2119 002664' 350 00 0 17 000000      AOS      (P)      ;Ret+2
2120 002665' 332 00 0 00 005705'      SKIPE      TRACE      ;Trace enabled?
2121 002666' 260 17 0 00 002635*      $TEXT      (<,.SSSTS successful>)      ;Yes
2122
2123 002670' 263 17 0 00 000000      SSSTSX: $RET
2124
2125      ; ARGUMENT BLOCK
2126
2127 002671' 000000 000000      STSARG: 0      ;Ret #wds processed,,block length
2128 002672' 000000 000000      0      ;CID
2129 002673' 000000 000000      0      ;Ret flags,,connect state code
2130 002674' 000000 000000      0      ;reserved,,node#

```

```

2131
2132      ; * SSRMG - Receive Message
2133      ;
2134      ; Data received is placed in a user's buffer space previously queued.
2135      ; SCS returns the address of the buffer in .SQARB of the jsys arg. block
2136      ;
2137      ; Return+1      if jsys error
2138      ; Return+2      if successful
2139
2140 002675' 304 00 0 00 000000      SSRMG: $SAVE    <S1,S2>      ;Save some ac's
2141 002702' 201 02 0 00 002717'      MOVEI    S2,RMGARG      ;Arg block addr
2142
2143 002703' 201 01 0 00 000005      MOVEI    S1,.LBRMG      ;Arg block length
2144 002704' 202 01 0 02 000000      MOVEM    S1,.SQLEN(S2)    ;Put in arg block
2145 002705' 200 01 0 00 005701'      MOVE     S1,CID
2146 002706' 202 01 0 02 000001      MOVEM    S1,.SQCID(S2)    ; " CID
2147
2148 002707' 201 01 0 00 000013      MOVEI    S1,.SSRMG      ;Function code
2149 002710' 260 17 0 00 002751'      $CALL    $SCS          ;Do the JSYS
2150 002711' 254 00 0 00 002716'      JRST     SSRMGX        ; Error ret
2151 002712' 350 00 0 17 000000      AOS      (P)          ;Ret+2
2152 002713' 332 00 0 00 005705'      SKIPE    TRACE        ;Trace enabled?
2153 002714' 260 17 0 00 002666*      $TEXT    (<,.SSRMG successful>) ;Yes
2154
2155 002716' 263 17 0 00 000000      SSRMGX: $RET
2156
2157      ; ARGUMENT BLOCK
2158
2159 002717' 000000 000000      RMGARG: 0      ;Ret #wds processed,.block length
2160 002720' 000000 000000      0      ;CID or -1
2161 002721' 000000 000000      0      ;Ret addr of msg buffer
2162 002722' 000000 000000      0      ;Ret flags,.node#
2163 002723' 000000 000000      0      ;Ret len of msg

```

```

2164
2165      ; * SSGDE - Get Entry from Data Queue
2166      ;
2167      ; Return+1      if jsys error
2168      ; Return+2      if successful
2169
2170      002724' 304 00 0 00 000000      SSGDE: $SAVE    <S1,S2>      ;Save some ac's
2171      002731' 201 02 0 00 002746'      MOVEI      S2,GDEARG      ;Arg block addr
2172
2173      002732' 201 01 0 00 000003      MOVEI      S1,..LBGDE      ;Arg block length
2174      002733' 202 01 0 02 000000      MOVEM      S1,..SQLEN(S2)      ;Put in arg block
2175      002734' 200 01 0 00 005701'      MOVE       S1,CID
2176      002735' 202 01 0 02 000001      MOVEM      S1,..SQCID(S2)      ; " CID
2177
2178      002736' 201 01 0 00 000024      MOVEI      S1,..SSGDE      ;Function code
2179      002737' 260 17 0 00 002751'      $CALL      $SCS          ;Do the JSYS
2180      002740' 254 00 0 00 003014'      JRST       FLAGIT        ;Flag the error and return +1
2181      002741' 350 00 0 17 000000      AOS         (P)          ;Ret+2
2182      002742' 332 00 0 00 005705'      SKIPE      TRACE        ;Trace enabled?
2183      002743' 260 17 0 00 002714*      $TEXT      (<..SSGDE successful>) ;Yes
2184      002745' 263 17 0 00 000000      $RET
2185
2186      ; ARGUMENT BLOCK
2187
2188      002746' 000000 000000      GDEARG: 0      ;Ret #wds processed,,block length
2189      002747' 000000 000000      0      ;CID or -1
2190      002750' 000000 000000      0      ;Ret buff ID which completed DMA xfer
  
```

```

2191          SUBTTL SCS JSYS/UUO Execution
2192
2193          ;* $SCS - Do the actual SCS% jsys
2194
2195          ; Enter with S1 = JSYS function code
2196          ;                S2 = address of JSYS argument block
2197          ; Return+1      = if jsys error
2198          ; Return+2      = if successful
2199
2200 002751' 304 00 0 00 000000 $SCS: $SAVE <S1,S2,T1>
2201 002757' 124 01 0 00 005730' DMOVEM S1,SCSFCN ;Save SCS function code, arg block adr
2202 002760' 250 02 0 17 000000 EXCH S2,(P) ;Get return PC, save S2
2203 002761' 552 02 0 00 005726' HRRZM S2,SCSCAL ;Save calling PC
2204 002762' 250 02 0 17 000000 EXCH S2,(P) ;Save return PC, get S2
2205 002763' 336 00 0 00 002263* SKIPN MONTYP ;TOPS-20?
2206 002764' 254 00 0 00 002774' JRST SCS$1 ;No, TOPS-10
2207
2208          ;TOPS-20: Execute SCS% JSYS, if error get it and save it, then return.
2209
2210 002765' 104 00 0 00 000622 SCS% ;THE JSYS
2211 002766' 320 16 0 00 002770' ERJMP +2 ;Ret+1 if error
2212 002767' 254 00 0 00 003012' JRST $SCSX ;Ret+2 if success
2213 002770' 201 01 0 00 400000 MOVEI S1,.FHSLF ;Load fork handle of self
2214 002771' 104 00 0 00 000012 GETER% ;Get the error code
2215 002772' 552 02 0 00 005727' HRRZM S2,SCSERC ;Save SCS error code
2216 002773' 263 17 0 00 000000 $RET ;Return+1: error
2217
2218          ;Here for TOPS-10. The SCS UUO wants length,,functioncode in the first word
2219          ; of the block were the SCS JSYS wants 0,,length. Swap the arguments around.
2220
2221 002774' 500 01 0 02 000000 SCS$1: HRL S1,(S2) ;Get length,,functioncode
2222 002775' 202 01 0 02 000000 MOVEM S1,(S2) ;Save it back in the argument block
2223 002776' 200 01 0 00 000002 MOVE S1,S2 ;Copy the address of the block
2224 002777' 047 01 0 00 000213 SCS S1, ;The UUO
2225 003000' 304 00 0 00 000000 CAIA ;Error return
2226 003001' 350 00 0 17 000000 AOS (P) ;Give skip return
2227 003002' 202 01 0 00 005727' MOVEM S1,SCSERC ;Save error code
2228 003003' 302 01 0 00 000013 CAIE S1,SSIC1% ;Invalid connect ID?
2229 003004' 263 17 0 00 000000 $RET ;No, return
2230 003005' 120 01 0 00 005730' DMOVE S1,SCSFCN ;Load the function code
2231 003006' 302 01 0 00 000012 CAIE S1,.SSSTS ;Was it get status?
2232 003007' 263 17 0 00 000000 $RET ;No, return
2233 003010' 201 01 0 00 000001 MOVX S1,SQ%CLD ;Load closed state
2234 003011' 202 01 0 02 000002 MOVEM S1,.SQFST(S2) ;Store this as the state
2235
2236 003012' 350 00 0 17 000000 $SCSX: AOS (P) ;Ret+2
2237 003013' 263 17 0 00 000000 $RET
  
```

```

2238
2239      ;* FLAGIT - Print error codes from SCS JSYS/UUO.
2240      ;Call:  MOVE    S1,[ASCIZ/XXX/]      ;For function .SSxxx
2241      ;        $CALL  FLAGIT                ;To print error message
2242      ;
2243
2244
2245 003014' 260 17 0 00 001772* FLAGIT: $SAVE <T1,T2,T3,T4>      ;Save some ACs
2246 003015' 336 00 0 00 002763*      SKIPN  MONTYP          ;TOPS-20?
2247 003016' 254 00 0 00 003022*      JRST   FLAGI1          ;No, TOPS-10
2248
2249 003017' 260 17 0 00 002743*      $TEXT  (,<? SCS% JSYS error: ^E/SCSERC/>)
2250 003021' 254 00 0 00 003031*      JRST   FLAGI4          ;Output more stuff
2251
2252 003022' 205 04 0 00 777743      FLAGI1: MOVSI  T2,-ERRSCL      ;Load -ive length of table,,0
2253 003023' 554 03 0 04 003057*      FLAGI2: HLRZ   T1,ERRSCT(T2)      ;Load error code
2254 003024' 312 03 0 00 005727*      CAME    T1,SCSERC      ;Match?
2255 003025' 253 04 0 00 003023*      AOBJN   T2,FLAGI2      ;Nope, loop
2256 003026' 550 04 0 04 003057*      HRRZ    T2,ERRSCT(T2)      ;Load address of ASCIZ text
2257 003027' 260 17 0 00 003017*      $TEXT  (,<? SCS UUO error: (^0/SCSERC/) ^T/(T2)/>)
2258
2259 003031' 370 00 0 00 005726*      FLAGI4: SOS    SCSCAL      ;Set true return PC
2260 003032' 205 06 0 00 777765      MOVSI  T4,-ERRSSL      ;Load table length
2261 003033' 554 05 0 06 003043*      FLAGI5: HLRZ   T3,ERRSST(T4)      ;Load the function code
2262 003034' 312 05 0 00 005730*      CAME    T3,SCSFCN      ;Match last function?
2263 003035' 253 06 0 00 003033*      AOBJN   T4,FLAGI5      ;No, keep looking
2264 003036' 550 06 0 06 003043*      HRRZ    T4,ERRSST(T4)      ;Point to the ASCIZ
2265 003037' 336 00 0 00 003015*      SKIPN   MONTYP          ;TOPS-20?
2266      $TEXT  (,< Function ^0/SCSFCN/: ^T/(T4)/
2267 003040' 260 17 0 00 003027*      arg block at ^0/SCSBLK/, called from PC ^0/SCSCAL/>)
2268 003042' 260 17 0 00 000000      $RET          ;Return to calling
  
```

2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318

003043' 000020 011424'
003044' 000000 011433'
003045' 000010 011440'
003046' 000003 011446'
003047' 000025 011454'
003050' 000024 011462'
003051' 000007 011470'
003052' 000011 011477'
003053' 000013 011506'
003054' 000006 011513'
003055' 000012 011520'
003056' 777777 011527'
000013
003057' 000000 011531'
003060' 000001 011534'
003061' 000002 011541'
003062' 000003 011546'
003063' 000004 011555'
003064' 000005 011563'
003065' 000006 011570'
003066' 000007 011576'
003067' 000010 011604'
003070' 000011 011611'
003071' 000012 011614'
003072' 000013 011621'
003073' 000014 011625'
003074' 000015 011631'
003075' 000016 011641'
003076' 000017 011646'
003077' 000020 011654'
003100' 000021 011663'
003101' 000022 011670'
003102' 000023 011675'
003103' 000024 011701'
003104' 000025 011706'
003105' 000026 011712'
003106' 000027 011721'
003107' 000030 011725'
003110' 000031 011731'
003111' 000032 011741'
003112' 000033 011745'
003113' 000034 011753'
003114' 777777 011763'
000035

;Table of SCS JSYS/UUO functions used in this program

ERRSST: XWD .SSAIC,[ASCIZ/.SSAIC (Attach interrupt channels)/]
 XWD .SSCON,[ASCIZ/.SSCON (Connect to node)/]
 XWD .SSCSP,[ASCIZ/.SSCSP (Connect state poll)/]
 XWD .SSDIS,[ASCIZ/.SSDIS (Disconnect from node)/]
 XWD .SSEVT,[ASCIZ/.SSEVT (Event queue read)/]
 XWD .SSGDE,[ASCIZ/.SSGDE (Get data queue entry)/]
 XWD .SSQRM,[ASCIZ/.SSQRM (Queue message buffers)/]
 XWD .SSRCD,[ASCIZ/.SSRCD (Return connection data)/]
 XWD .SSRMG,[ASCIZ/.SSRMG (Read message)/]
 XWD .SSSMG,[ASCIZ/.SSSMG (Send message)/]
 XWD .SSSTS,[ASCIZ/.SSSTS (Get connection status)/]
 XWD 777777,[ASCIZ/(unknown)/]
 ERRSSL==.-ERRSST-1 ;Number of 'real' SCS functions

;Table of SCS UUO error codes

ERRSCT: XWD SSNPV%,[ASCIZ/Not privileged/]
 XWD SSIFC%,[ASCIZ/Illegal function code/]
 XWD SSARG%,[ASCIZ/Bad argument list length/]
 XWD SSACR%,[ASCIZ/Address check reading arguments/]
 XWD SSACS%,[ASCIZ/Address check storing data/]
 XWD SSCPN%,[ASCIZ/CPU number out of range/]
 XWD SSNPC%,[ASCIZ/No CI port on specified CPU/]
 XWD SSNNK%,[ASCIZ/CPU's node number not known/]
 XWD SSINN%,[ASCIZ/Invalid CI node number/]
 XWD SSNFC%,[ASCIZ/No free core/]
 XWD SSVNO%,[ASCIZ/Virtual circuit not open/]
 XWD SSICI%,[ASCIZ/Invalid connect id/]
 XWD SSRQE%,[ASCIZ/Receive queue empty/]
 XWD SSNBQ%,[ASCIZ/No buffer queued for packet reception/]
 XWD SSRCF%,[ASCIZ/Reject connection failed/]
 XWD SSDCF%,[ASCIZ/Disconnect connection failed/]
 XWD SSNFB%,[ASCIZ/No free buffers to send packet/]
 XWD SSQBF%,[ASCIZ/Queue buffers failed/]
 XWD SSCBF%,[ASCIZ/Cancel buffers failed/]
 XWD SSPSF%,[ASCIZ/Package send failed/]
 XWD SSDQE%,[ASCIZ/Data entry queue empty/]
 XWD SSEQE%,[ASCIZ/Event queue empty/]
 XWD SSCRB%,[ASCIZ/Can't remove buffer from database/]
 XWD SSCUB%,[ASCIZ/Can't unmap buffer/]
 XWD SSNSB%,[ASCIZ/No such buffer name/]
 XWD SSTMS%,[ASCIZ/Too many buffer segment descriptors/]
 XWD SSIDM%,[ASCIZ/Illegal data mode/]
 XWD SSSCP%,[ASCIZ/Segment crosses page boundary/]
 XWD SSSTL%,[ASCIZ/Segment too long (greater than 1 page)/]
 XWD 777777,[ASCIZ/Unknown error code/]
 ERRSCL==.-ERRSCT-1 ;Number of 'real' errors

SUBTTL Spear Reporting Subroutines

; * SPR1 - Send initial spear report 'diagnostic started'

; This subroutine makes a 'DIAGNOSTIC STARTED' spear report. It determines if
 ; it should be made by looking at 'SPEAR'. 'SPEAR' is a flag word that
 ; is toggled by the commands ENABLE/DISABLE SPEAR-REPORTING. Spear reporting
 ; is never enabled on TOPS-10 because there is no facility to do it.

SPR1: \$SAVE <S2,S1> ;Save some ac's
 SKIPE MONTYP ;Skip if TOPS-10
 SKIPN SPEAR ;Spear reporting enabled ?
 \$RET ;Bypass spear report

; Build entry

\$CALL I%NOW ;Get time and day
 MOVEM S1,SPR1TI ;Save it
 \$TEXT (<-1,,SPR1MS>,<DFCID version ^V/.JBVER/ user ^U/[-1]/ connecting to node ^D/
 NODNUM/.^0>)

; Make the entry

MOVEI S1,SPR1BL ;Get address of argument block
 MOVEI S2,SPR1LN ;Get length
 SYERR% ;Write to syserr file
 ERJMP .+2 ;Report error
 \$RET ;Return
 \$TEXT (,<? Can't write spear entry at connect: ^E/[-1]/>)
 \$RET ;Return

; Spear entry

SPR1BL: 250000,.0 ;Header: Event type 250
 0 ;Zero
 0 ;Zero
 0 ;Zero
 0 ;Zero
 2B23+2 ;Time of occurrence block
 SPR1TI: 0 ;Time
 14B23+SPR1LN-6 ;Comment code, block length
 2 ;String offset
 SPR1MS: BLOCK ^D90/5 ;Should be plenty
 SPR1LN=.-SPR1BL ;Length

2319
 2320
 2321
 2322
 2323
 2324
 2325
 2326
 2327
 2328 003115' 304 00 0 00 000000
 2329 003122' 332 00 0 00 003037*
 2330 003123' 336 00 0 00 005704'
 2331 003124' 263 17 0 00 000000
 2332
 2333
 2334
 2335 003125' 260 17 0 00 000000*
 2336 003126' 202 01 0 00 003146'
 2337
 2338 003127' 260 17 0 00 003040*
 2339
 2340
 2341
 2342 003131' 201 01 0 00 003141'
 2343 003132' 201 02 0 00 000032
 2344 003133' 104 00 0 00 000527
 2345 003134' 320 16 0 00 003136'
 2346 003135' 263 17 0 00 000000
 2347 003136' 260 17 0 00 003127*
 2348 003140' 263 17 0 00 000000
 2349
 2350
 2351
 2352 003141' 250000 000000
 2353 003142' 000000 000000
 2354 003143' 000000 000000
 2355 003144' 000000 000000
 2356 003145' 000000 020002
 2357 003146' 000000 000000
 2358 003147' 000000 140024
 2359 003150' 000000 000002
 2360 003151' 000000
 2361 000032

```

2362
2363      ;* SPR2 - makes successful completion spear report.
2364
2365      003173' 304 00 0 00 000000      SPR2:  $SAVE    <S1,S2>          ;Save some ac's
2366      003200' 332 00 0 00 003122*      $SKIPE    MONTP      ;Skip if TOPS-10
2367      003201' 336 00 0 00 005704'      $SKIPN    SPEAR      ;Spear reporting enabled ?
2368      003202' 263 17 0 00 000000      $RET          ;It isn't. Exit this subroutine
2369      003203' 260 17 0 00 003125*      $CALL     I%NOW      ;Get time and day
2370      003204' 202 01 0 00 003224'      MOVEM     S1,SPR2TI    ;Save it
2371      003205' 260 17 0 00 003136*      $TEXT     (<-1,,SPR2MS>,<DFCID version ^V/.JBVER/ user ^U/[-1]/ disconnect from node ^
2372      003207' 201 02 0 00 000032      D/NODNUM/.^0>)
2373      003210' 201 01 0 00 003217'      MOVEI     S2,SPR2LN      ;Load length
2374      003211' 104 00 0 00 000527      MOVEI     S1,SPR2BL      ;Get address of argument block
2375      003212' 320 16 0 00 003214'      SYERR%      ;Write to syserr file
2376      003213' 263 17 0 00 000000      ERJMP     .+2          ;Report error
2377      003214' 260 17 0 00 003205*      $RET          ;
2378      003216' 263 17 0 00 000000      $TEXT     (,<? Can't write spear entry at disconnect: ^E/[-1]/>)
2379      003216' 263 17 0 00 000000      $RET          ;
2380
2381      ; Spear entry block
2382
2383      003217' 250000 000000      SPR2BL: 250000,,0          ;Event type
2384      003220' 000000 000000      0          ;Zero
2385      003221' 000000 000000      0          ;Zero
2386      003222' 000000 000000      0          ;Zero
2387      003223' 000000 020002      2B23+2          ;Time of occurrence block type 2
2388      003224' 000000 000000      SPR2TI: 0          ;Time
2389      003225' 000000 140024      14B23+SPR2LN-6      ;Type 11 code, block length
2390      003226' 000000 000002      2          ;String offset
2391      003227'          000032      SPR2MS: BLOCK ^D90/5      ;Should be big enough
2392          SPR2LN==.-SPR2BL          ;Length of block

```

SUBTTL Message Buffers

; * MESSAGE BUFFER CHAIN - Used only for RECEIVING Messages from the DUP Server
 ;
 ; These buffers are initially queued in .SSCON (Connect).
 ; If a MESSAGE AVAILABLE occurs, a .SSRMG (Receive Msg) is performed and the
 ; particular buffer pointed to is read.
 ; When finished, the buffer is re-queued by setting the first location to zero,
 ; putting that buffer address in the .SSQRM (Queue Msg Buffer) arg block
 ; and performing a .SSQRM

2393					
2394					
2395					
2396					
2397					
2398					
2399					
2400					
2401					
2402					
2403					
2404	003251'	000000	003325'	MBUF1:	MBUF2
2405	003252'				BLOCK
2406					^D43
2407	003325'	000000	003401'	MBUF2:	MBUF3
2408	003326'				BLOCK
2409					^D43
2410	003401'	000000	003455'	MBUF3:	MBUF4
2411	003402'				BLOCK
2412					^D43
2413	003455'	000000	003531'	MBUF4:	MBUF5
2414	003456'				BLOCK
2415					^D43
2416	003531'	000000	003605'	MBUF5:	MBUF6
2417	003532'				BLOCK
2418					^D43
2419	003605'	000000	003661'	MBUF6:	MBUF7
2420	003606'				BLOCK
2421					^D43
2422	003661'	000000	003735'	MBUF7:	MBUF8
2423	003662'				BLOCK
2424					^D43
2425	003735'	000000	004011'	MBUF8:	MBUF9
2426	003736'				BLOCK
2427					^D43
2428	004011'	000000	004065'	MBUF9:	MBUF10
2429	004012'				BLOCK
2430					^D43
2431	004065'	000000	004141'	MBUF10:	MBUF11
2432	004066'				BLOCK
2433					^D43
2434	004141'	000000	004215'	MBUF11:	MBUF12
2435	004142'				BLOCK
2436					^D43
2437	004215'	000000	000000	MBUF12:	0
2438	004216'				BLOCK

;link

;end


```

2465          SUBTTL Storage
2466
2467          ;EVENT FLAGS asserted when -1
2468
2469          005654' 000000 005654' EFLST=. ;Start of event flags
2470          005654' 000000 000000 CONACC: 0 ;Connection accepted
2471          005655' 000000 000000 SDONE: 0 ;Msg/DG send complete
2472
2473          005656' 000000 000000 EQEMTY: 0 ;Event Queue Empty
2474          005657' 000000 000000 MAVAIL: 0 ;Message available
2475          005660' 000000 000000 EVTCNT: 0 ;Event count (for debug only)
2476          005660' 005660' EFLLEN=-1 ;Event flag table length
2477
2478          ;Flags returned in MSGSRV
2479
2480          005661' 000000 000000 HIDDEN: 0 ;-1 if Hi density mode
2481          005662' 000000 000000 BADNOD: 0 ;-1 if ret node# not same as desired
2482
2483          ;Info returned in the GET DUST STATUS response
2484
2485          005663' 000000 000000 PGMEXT: 0 ;Ascii EXT for down-line loadable pgms
2486          005664' 000000 000000 PRGIND: 0 ;Progress indicator
2487          005665' 000000 000000 LPTMO: 0 ;Timeout for EXEC LOCAL PGM cmd in sec
2488          005666' 000000 000000 DISSRV: 0 ;-1 if pgm execution DISABLES all svrs
2489          005667' 000000 000000 LODMED: 0 ;-1 if contr HAS local load media
2490          005670' 000000 000000 NOEXSP: 0 ;-1 if server CANT do EXEC SUPP PGM cmd
2491          005671' 000000 000000 ACTIVE: 0 ;-1 if server currently in ACTIVE state
2492          005672' 000000 000000 SNDDIM: 0 ;-1 if contr SUPPORTS Send Data Immed
2493
2494          ;Info returned in the EXECUTE LOCAL PGM response
2495
2496          005673' 000000 000000 PGMVER: 0 ;Program version
2497          005674' 000000 000000 DATIMO: 0 ;Timeout for send-recv data and SDI
2498          005675' 000000 000000 STANDALONE: 0 ;-1 if program is STAND-ALONE
2499          005676' 000000 000000 OVERLA: 0 ;-1 if pgm REQ OVERLAYS from host
2500          005677' 000000 000000 WRITEB: 0 ;-1 if overlays should be WRITEABLE
2501          005700' 000000 000000 STDIAL: 0 ;-1 if pgm uses STD DUP DIALOGUE
  
```


DFCIDM DUF Driver Main Module version 1(12) MACRO %53B(1242) 12:08 6-Sep-85 Page 67
DFCIDM MAC 6-Sep-85 10:19 Literals

SEQ 0156

```
2534          SUBTTL Literals
2535
2536      ;Literals dumped here
2537
2538 005732'    LIT..M: XLIST          ;LIT
2539           LIST
2540
2541           END
```

NO ERRORS DETECTED

PROGRAM BREAK IS 012147
CPU TIME USED 02:50.333

168P CORE USED

CONN1	757	764#					
CONN2	760	767#					
CONNEC	162	746#					
CONNX	753	762	765	771#			
CONX	157	163	222#				
CSPARG	2077	2095#					
CUOPC	663	1579	1618	1663	1709	1784	2525#
CYCL60	1243	1249	1258	1261#			
DATTMO	860	929	2497#				
DBUF1	1436	1909	2450#				
DBUF2	1435	1438	2450	2453#			
DBUF3	1437	1440	2453	2456#			
DBUF4	1439	1442	2456	2459#			
DBUF5	1441	1443	2459	2462#			
DECVER	106						
DGMCHN	250#	1829					
DGMLEV	250#						
DGSRV	249	481#					
DISARG	1979	1998#					
DISCON	221	1001#					
DISFLG	211	309	362	429	1294	2518#	
DISSRV	1082	1095	2488#				
DLTIM	1221	1228	1275	1282	2512#		
DMACHN	247#	1826					
DMALEV	247#						
DMSRV	246	476#					
DPN	1901	1938#					
DSC4	1006	1013#					
DSTRT	1224	1227	1278	1281	2513#		
DUST1	1043	1049#					
DUST2	1045	1052#					
DUST4	1056	1059#					
DUST5	1057	1062#					
DUSTAT	167	449	1033#				
DUSTST	1038	1573#					
DUSTX	1039	1047	1050	1060	1064#		
EFLN	1406	2476#					
EFLST	1404	1405	2469#				
EML1	798	805#					
EML11	846	850#					
EML13	832	851	856#				
EML15	863	866#					
EML2	801	808#					
EML4	813	816#					
EML5	814	820#					
EML7	836	840#					
EML9	841	845#					
EMLX	794	803	806	818	821	867#	
EMULAT	182	787#					
EQEMTY	551	790	889	959	1230	2473#	
ERRSCL	2252	2318#					
ERRSCT	2253	2256	2288#	2318			
ERRSSL	2260	2284#					

ERRSST	2261	2264	2272#	2284															
ETIME	250	437#																	
EVNTBL	539	556#																	
EVTARG	518	596	1946	1964#															
EVTCHN	248#	1827																	
EVCNT	2475#																		
EVTLEV	248#																		
EVTSO	486	503	511#																
EVTST	515#	540																	
EVTST3	522	539#																	
EVTST4	516	544#																	
EVTSRV	247	486#																	
EVW2	1226#	1231																	
EVWAIT	756	797	915	966	1042	1220#													
EVWX	1229	1233#																	
EXLBLK	1615	1620	1622	1631#	1636														
EXLLEN	1623	1636#																	
EXLPGM	793	1612#																	
EXPINV	1460	1469	1858	2514#															
FLAGI1	2247	2252#																	
FLAGI2	2253#	2255																	
FLAGI4	2250	2259#																	
FLAGI5	2261#	2263																	
FLAGIT	548	730	1013	1817	1857	1859	1914	1990	2023	2058	2087	2180	2245#						
GDEARG	2171	2188#																	
GETTI1	1239	1243#																	
GETTI2	1244	1252#																	
GETTIM	1223	1226	1237#	1277	1280														
HIDEN	2480#																		
IXIOFF	1352																		
IXION	1328	1347																	
IXNOW	2335	2369																	
IBUFCT	188	332	896	906	1292	2520#													
IGNORE	556	564	565	568	569	573#													
INIT1	193	1291#																	
INITI1	459	468#																	
INITIM	457#	1296																	
INVRES	667																		
K%XTI	325																		
KEEPCN	437	442	447#																
LEV TAB	143	254#	312	414															
LIT..M	2538#																		
LOCRED	562	615#																	
LODMED	170	1083	1105	2489#															
LOGIFN	147#	902																	
LOOP	200#	213																	
LOOP1	206	210#	311	413	430														
LOOP2	202	208#																	
LOOPX	168	171	173	183	191	212	217#												
LOUT%	148#	903																	
LPGTMO	811	1149	1151	2487#															
LSTIN.	138	181	225	299	299#	301	368	411	411#	423	423#	429	449	451					
	458	458#	466	471	477	482	500	507	512	512#	514	538	551	578					

SEQ 0159

	583	590	593	593#	598	600	602	604	606	608	613	618	623	638
	638#	640	655	669	671	681	708	715	722	751	762	765	769	789
	803	806	810	818	835	839	844	849	855	866	886	886#	888	906
	921	924	928	936	958	971	974	978	984	1002	1002#	1004	1011	1015
	1034	1034#	1037	1047	1050	1054	1060	1082	1082#	1090	1098	1103	1109	1115
	1122	1126	1132	1135	1161	1184	1184#	1190	1206	1209	1221	1221#	1238	1238#
	1272	1272#	1292	1292#	1319	1319#	1335	1338	1341	1344	1347	1352	1352#	1361
	1364	1367	1370	1373	1404	1404#	1458	1458#	1459	1477	1483	1511	1511#	1521
	1521#	1557	1574	1574#	1613	1613#	1658	1658#	1704	1704#	1740	1740#	1779	1779#
	1811	1811#	1820	1839	1839#	1853	1895	1895#	1920	1946	1946#	1959	1979	1979#
	1994	2011	2011#	2027	2046	2046#	2062	2077	2077#	2091	2109	2109#	2122	2141
	2141#	2154	2171	2171#	2184	2201	2201#	2246	2250	2258	2268	2329	2329#	2339
	2348	2366	2366#	2373	2379									
	732	791	890	960	1284	2474#								
MAVAIL	1280#	1285												
MAW2	812	930	979	1055	1271#									
MAWAIT	1283	1287#												
MAWX	1412	1907	2404#											
MBUF1	1427	1430	2428	2431#										
MBUF10	1429	1432	2431	2434#										
MBUF11	1431	1433	2434	2437#										
MBUF12	1411	1414	2404	2407#										
MBUF2	1413	1416	2407	2410#										
MBUF3	1415	1418	2410	2413#										
MBUF4	1417	1420	2413	2416#										
MBUF5	1419	1422	2416	2419#										
MBUF6	1421	1424	2419	2422#										
MBUF7	1423	1426	2422	2425#										
MBUF8	1425	1428	2425	2428#										
MBUF9	147#	201	351	393	458	1238	1297	1319	1353	1808	2205	2246	2265	2329
MONTYP	2366													
MSGCHN	249#	1828												
MSGLEV	249#													
MSGSR0	491	505	637#											
MSGSRV	248	491#												
MSRV10	655	671	681	686	706#									
MSRV12	641#	722												
MSRV16	642	726#												
MSRV2	695#	697	699											
MSRV3	695	701#												
MSRV4	651	659#												
MSRV6	666	675#												
MSRV7	662	690#												
MSRV8	678	685#												
MSRVX	716	719	732#											
NODNUM	143	650	1511	1903	2506#									
NOEXSP	1084	1112	2490#											
OFFLIN	563	620#												
ONCHNS	230#	246	246#	247	247#	248	248#	249	249#	250	250#	251	251#	252
	252#	1326	1381											
OPCABT	1783													
OPCELP	1617													
OPCGDS	1578													

SEQ 0160

OPCRES	664	1743												
OPCSD	1662													
OPCSDI	661	1708	1743											
OVERLA	826	842	2499#											
P	139	200	210	299	411	423	458	512	593	638	770	866	886	940
	988	1002	1034	1063	1082	1184	1221	1232	1238	1272	1286	1292	1319	1352
	1404	1458	1511	1521	1574	1587	1613	1626	1658	1671	1704	1717	1740	1751
	1779	1792	1811	1820	1839	1850	1895	1920	1946	1956	1979	1991	2011	2024
	2046	2059	2077	2088	2109	2119	2141	2151	2171	2181	2201	2202	2204	2226
	2236	2329	2366											
P1	298	410	422	457	499	500	502	504	511	592	637	885	1001	1033
	1081	1183	1220	1237	1271	1291	1318	1351	1403	1457	1483	1510	1520	1573
	1612	1657	1703	1739	1778	1810	1838	1894	1945	1978	2010	2045	2076	2108
P2	2140	2170	2200	2246	2328	2365								
	298	410	422	457	499	511	592	637	885	1001	1033	1081	1183	1220
	1237	1271	1291	1318	1351	1403	1457	1483	1510	1520	1573	1612	1657	1703
	1739	1778	1810	1838	1894	1945	1978	2010	2045	2076	2108	2140	2170	2200
P3	2246	2328	2365											
	298	410	422	457	499	511	592	637	885	1001	1033	1081	1183	1220
	1237	1271	1291	1318	1351	1403	1457	1483	1510	1520	1573	1612	1657	1703
	1739	1778	1810	1838	1894	1945	1978	2010	2045	2076	2108	2140	2170	2200
P4	2246	2328	2365											
	298	410	422	457	499	511	592	637	885	1001	1033	1081	1183	1220
	1237	1271	1291	1318	1351	1403	1457	1483	1510	1520	1573	1612	1657	1703
	1739	1778	1810	1838	1894	1945	1978	2010	2045	2076	2108	2140	2170	2200
	2246	2328	2365											
P4A8	1475	1482#	1523	1527	1531	1534	1538							
P4A81	1488#	1490												
PC1	254	258#												
PC2	255	259#												
PC3	256	260#												
PCRLF	148#	347	1209											
PDISCN	567	628#												
PGMEXT	1137	2485#												
PGMVER	857	2486#												
PIVECT	143	264#	264	269	274	279	313	423	431	500				
PRGIND	1140	1142	1144	1146	2486#									
PRRCD1	1465	1474#												
PRRCD2	1514	1520#												
PS.FAC	1332	1335	1338	1341	1344									
PS.FRC	426	1358	1361	1364	1367	1370								
PS.RIA	272	426	1341	1367										
PS.RID	272	426	1341	1367										
PS.TMS	468													
QRMARG	2046	2066#												
RCDARG	1474	1522	1526	1530	1533	1537	1540	1839	1864#					
RD%BTM	335													
RD%JFN	373													
RDISCN	566	625#												
REJECT	560	592#												
REJX	595	608#												
RESTIM	462	468												
RESTO2	1354	1375#												

SEQ 0161

RESTOR	222	1351#												
RESTX	1387	1398#												
RMGARG	646	649	2141	2159#	203	204	298	311	312	313	318	319	320	321
S1	185	186	187	188	331	332	334	335	340	341	342	344	346	348
	322	323	324	330	422	426	427	430	431	457	461	468	469	511
	350	396	410	411	539	592	596	598	600	602	604	606	637	646
	518	519	520	535	664	665	675	676	677	690	691	696	697	717
	647	649	650	663	831	835	840	845	850	856	857	859	860	885
	718	755	796	811	914	929	965	978	1001	1033	1041	1054	1081	1092
	896	897	900	901	1127	1136	1137	1139	1140	1141	1142	1143	1144	1145
	1093	1099	1110	1117	1151	1155	1183	1185	1186	1192	1193	1203	1207	1220
	1146	1148	1149	1150	1245	1246	1247	1248	1249	1252	1253	1254	1255	1256
	1221	1224	1227	1228	1273	1274	1275	1278	1281	1282	1291	1300	1307	1310
	1257	1258	1271	1272	1332	1333	1335	1336	1338	1339	1341	1342	1344	1345
	1318	1321	1323	1325	1362	1364	1365	1367	1368	1370	1371	1375	1377	1380
	1351	1358	1359	1361	1406	1411	1412	1413	1414	1415	1416	1417	1418	1419
	1389	1394	1403	1405	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433
	1420	1421	1422	1423	1439	1440	1441	1442	1443	1474	1483	1520	1522	1526
	1435	1436	1437	1438	1541	1542	1543	1544	1546	1810	1812	1813	1815	1838
	1530	1533	1537	1540	1847	1894	1897	1898	1899	1900	1901	1902	1903	1904
	1841	1842	1843	1844	1909	1910	1912	1916	1917	1945	1948	1949	1950	1951
	1905	1906	1907	1908	1983	1984	1985	1986	1988	2010	2013	2014	2015	2016
	1953	1978	1981	1982	2050	2051	2052	2053	2054	2056	2076	2079	2080	2081
	2021	2045	2048	2049	2112	2113	2114	2116	2140	2143	2144	2145	2146	2148
	2082	2085	2108	2111	2176	2178	2200	2201	2213	2221	2222	2223	2224	2227
	2170	2173	2174	2175	2234	2328	2336	2342	2365	2370	2374			
	2228	2230	2231	2233	457	462	511	523	544	545	546	637	660	661
	298	410	422	457	727	728	885	1001	1081	1156	1183	1194	1206	1220
	667	692	695	726	1303	1309	1311	1318	1326	1351	1381	1390	1395	1484
	1237	1271	1291	1302	1532	1535	1810	1811	1813	1838	1839	1842	1844	1845
	1485	1520	1524	1528	1898	1900	1902	1904	1906	1908	1910	1945	1946	1949
	1855	1856	1894	1895	1984	1986	2010	2011	2014	2016	2017	2018	2019	2045
	1951	1978	1979	1982	2076	2077	2080	2082	2083	2108	2109	2112	2114	2140
	2046	2049	2051	2054	2171	2174	2176	2200	2202	2203	2204	2215	2221	2222
	2141	2144	2146	2170	2365	2373								
	2223	2234	2328	2343										
	200	210	2527#											
SAVEP	502													
SC%EVA	504	718												
SC%MSA	156	1808#												
SCAINT	1809	1820#												
SCAINTX	264#	500	1332	1358										
SCOFFS	2206	2221#												
SCS\$1	2532#													
SCSBLK	2203	2259	2529#											
SCSCAL	544	726	1855	2215	2227	2254	2530#							
SCSERC	2201	2230	2262	2531#										
SCSFCN	265	496#												
SCSI10	496	499#												
SCSI11	1856													
SCSISB	545	727												
SCSQIE	916	923#												
SDAT1	919	926#												
SDAT2														

S2

SEQ 0163

SDAT4	931	934#							
SDAT5	932	938#							
SDATX	910	921	924	936	939	941#			
SDCBLK	894	1706	1711	1713	1722#	1727			
SDCLEN	1714	1727#							
SDIREF	691	1740	2516#						
SDONE	610	789	800	888	918	958	968	1044	2471#
SDRBLK	1741	1745	1747	1756#	1759				
SDRLEN	1748	1759#							
SEND	1657#								
SENDIC	909	1703#							
SENDIR	701	1739#							
SMGARG	2011	2031#							
SND1	900#	907							
SNDBLK	1660	1665	1667	1676#	1683				
SNDDIM	172	1086	1129	2492#					
SNDHSC	190	356	885#						
SNDLEN	1668	1683#							
SNET	142	1457#							
SNET1	1463#	1468							
SNOD	142	1510#							
SNODX	1513	1515#							
SPEAR	143	2330	2367	2509#					
SPN	1899	1937#							
SPR1	769	2328#							
SPR1BL	2342	2352#	2361						
SPR1LN	2343	2358	2361#						
SPR1MS	2360#								
SPR1TI	2336	2357#							
SPR2	1008	2365#							
SPR2BL	2374	2383#	2392						
SPR2LN	2373	2389	2392#						
SPR2MS	2391#								
SPR2TI	2370	2388#							
SQ% CLO	2233								
SSACR%	2291								
SSACS%	2292								
SSARG%	2290								
SSCBF%	2306								
SSCON	752	1894#							
SSCPN%	2293								
SSCRB%	2310								
SSCSP	2076#								
SSCUB%	2311								
SSDCF%	2303								
SSDIS	1005	1978#							
SSDQE%	2308								
SSEQE%	546	2309							
SSEVT	515	1945#							
SSEVTX	1955	1960#							
SSGDE	2170#								
SSICIX	2228	2299							
SSIDM%	2314								

SSIFC%	2289													
SSINN%	2296													
SSNBQ%	2301													
SSNFB%	2304													
SSNFC%	2297													
SSNNK%	2295													
SSNPC%	2294													
SSNPV%	2288													
SSNSB%	2312													
SSPSF%	2307													
SSQBF%	2305													
SSQRM	708	2045#												
SSRCD	1463	1512	1838#											
SSRCD1	1849	1855#												
SSRCF%	2302													
SSRMG	641	2140#												
SSRMGX	2150	2155#												
SSRQE%	728	2300												
SSSCP%	2315													
SSSMG	1585	1624	1669	1715	1749	1790	2010#							
SSSTL%	2316													
SSSTS	715	2108#												
SSSTSX	2118	2123#												
SSTMS%	2313													
SSVNO%	2298													
STANDA	825	837	2498#											
STDIAL	828	852	2501#											
STSARG	717	2109	2127#											
STSBLK	1576	1581	1583	1592#	1595									
STSLN	1584	1595#												
SYSIN1	1320	1332#												
SYSINT	155	1318#												
T%TEXT	180	224	300	367	428	448	450	465	470	476	481	506	513	537
	550	577	582	589	597	599	601	603	605	607	612	617	622	639
	654	668	670	680	707	714	721	750	761	764	768	788	802	805
	809	817	834	838	843	848	854	865	887	905	920	923	927	935
	957	970	973	977	983	1003	1010	1014	1036	1046	1049	1053	1059	1089
	1097	1102	1108	1114	1121	1125	1131	1134	1160	1189	1205	1208	1334	1337
	1340	1343	1346	1360	1363	1366	1369	1372	1458	1476	1556	1819	1852	1919
	1958	1993	2026	2061	2090	2121	2153	2183	2249	2257	2267	2338	2347	2372
	2378													
T1	413	414	457	463	637	659	660	675	690	692	693	830	831	856
	859	1081	1091	1092	1136	1139	1141	1143	1145	1148	1150	1155	1156	1157
T2	1183	1184	1185	1206	1482	1483	1488	1520	1524	2200	2245	2253	2254	
	637	885	894	897	898	901	1081	1157	1457	1461	1466	1467	1482	1484
	1489	1510	1511	1520	1528	1573	1574	1575	1576	1578	1579	1580	1581	1583
	1612	1613	1614	1615	1617	1618	1619	1620	1622	1657	1658	1659	1660	1662
	1663	1664	1665	1667	1703	1704	1705	1706	1708	1709	1710	1711	1713	1739
	1740	1741	1743	1744	1745	1747	1778	1779	1780	1781	1783	1784	1785	1786
	1788	1838	1845	2010	2017	2076	2083	2245	2252	2253	2255	2256		
T3	637	895	900	1291	1482	1488	1489	1520	1532	1573	1584	1612	1623	1657
	1668	1703	1714	1739	1748	1778	1789	2010	2018	2245	2261	2262		
T4	298	410	422	457	499	511	592	637	693	696	885	1001	1033	1081

	1183	1220	1237	1271	1291	1318	1351	1403	1457	1482	1486	1490	1510	1520
	1535	1573	1612	1657	1703	1739	1778	1810	1838	1894	1945	1978	2010	2045
	2076	2108	2140	2170	2200	2245	2260	2261	2263	2264	2328	2365		
TIMCHN	251#	463												
TIMI10	275	442#												
TIMLEV	251#													
TIOFFS	274#	1335	1361											
TRACE	143	299	447	512	521	549	576	580	586	594	611	615	621	638
	706	713	720	749	767	787	808	833	853	862	886	904	926	956
	976	1002	1009	1035	1052	1088	1106	1113	1120	1124	1130	1153	1188	1818
	1851	1918	1957	1992	2025	2060	2089	2120	2152	2182	2510#			
TT00	288	293	298#											
TT1	305	318#												
TT10	304#	343	358	363	368									
TT3	345	357	362#											
TT4	336	349	356#											
TT8	326	367#												
TTCHR1	394	401#												
TTCHRC	304	393#												
TTIBUF	186	318	376	379	382	895	2521#							
TTICHN	246#	1321												
TTILEV	246#	312												
TTINP	245	288#												
TTISIZ	322	330	378	1726	2521									
TTIWD	1309	1395	2524#											
TTMOD	1302	1386	1390	1398	2523#									
TTOFFS	269#	313	426	1338	1341	1364	1367							
TTYI10	270	293#												
TXDONE	561	610#												
TXTARG	143	319	320	321	323	324	331	334	340	372#				
VC.OPN	1542	1543												
VCBROK	557	575#												
VEDIT	106													
WRITEB	827	847	2500#											
\$RETF	139	399	402											
\$RETIF	139													
\$RETIT	139													
\$RETT	139	398	403											
\$SCS	1816	1848	1913	1954	1989	2022	2057	2086	2117	2149	2179	2200#		
\$SCSX	2212	2236#												
..0001	299	299#												
..0002	411	411#												
..0003	423	423#												
..0004	458	458#												
..0005	500#													
..0006	512	512#												
..0007	593	593#												
..0010	638	638#												
..0011	886	886#												
..0012	1002	1002#												
..0013	1034	1034#												
..0014	1082	1082#												
..0015	1184	1184#												

SEQ 0165

..0016	1221	1221#												
..0017	1238	1238#												
..0020	1272	1272#												
..0021	1292	1292#												
..0022	1319	1319#												
..0023	1352	1352#												
..0024	1404	1404#												
..0025	1458	1458#												
..0026	1483#													
..0027	1511	1511#												
..0030	1521	1521#												
..0031	1574	1574#												
..0032	1613	1613#												
..0033	1658	1658#												
..0034	1704	1704#												
..0035	1740	1740#												
..0036	1779	1779#												
..0037	1811	1811#												
..0040	1839	1839#												
..0041	1895	1895#												
..0042	1946	1946#												
..0043	1979	1979#												
..0044	2011	2011#												
..0045	2046	2046#												
..0046	2077	2077#												
..0047	2109	2109#												
..0050	2141	2141#												
..0051	2171	2171#												
..0052	2201	2201#												
..0053	2246#													
..0054	2329	2329#												
..0055	2366	2366#												
..MX1	468#	468	1311#	1311	1321#	1321	1323#	1323	1326#	1326	1381#	1381	2233#	2233
..MX2	468#	468	1311#	1312	1321#	1321	1322	1323#	1323	1326#	1326	1381#	1381	2233#
..TSA1	2234													
	298#	298	410#	410	422#	422	457#	457	499#	499	511#	511	592#	592
	637#	637	885#	885	1001#	1001	1033#	1033	1081#	1081	1183#	1183	1220#	1220
	1237#	1237	1271#	1271	1291#	1291	1318#	1318	1351#	1351	1403#	1403	1457#	1457
	1482#	1482	1510#	1510	1520#	1520	1573#	1573	1612#	1612	1657#	1657	1703#	1703
	1739#	1739	1778#	1778	1810#	1810	1838#	1838	1894#	1894	1945#	1945	1978#	1978
	2010#	2010	2045#	2045	2076#	2076	2108#	2108	2140#	2140	2170#	2170	2200#	2200
	2245#	2245	2328#	2328	2365#	2365								
..TX1	502#	502	504#	504										
..TX2	502#	502	503	504#	504	505								
..A13	298	410	422	457	500	511	592	637	885	1001	1033	1081	1183	1220
	1237	1271	1291	1318	1351	1403	1457	1483	1510	1520	1573	1612	1657	1703
	1739	1778	1810	1838	1894	1945	1978	2010	2045	2076	2108	2140	2170	2200
	2246	2328	2365											
..A14	298	410	422	457	500	511	592	637	885	1001	1033	1081	1183	1220
	1237	1271	1291	1318	1351	1403	1457	1483	1510	1520	1573	1612	1657	1703
	1739	1778	1810	1838	1894	1945	1978	2010	2045	2076	2108	2140	2170	2200
	2246	2328	2365											
..A15	298	410	422	457	500	511	592	637	885	1001	1033	1081	1183	1220

	1237	1271	1291	1318	1351	1403	1457	1483	1510	1520	1573	1612	1657	1703
	1739	1778	1810	1838	1894	1945	1978	2010	2045	2076	2108	2140	2170	2200
.A16	2246	2328	2365											
	298	410	422	457	499	500	511	592	637	885	1001	1033	1081	1183
	1220	1237	1271	1291	1318	1351	1403	1457	1483	1510	1520	1573	1612	1657
	1703	1739	1778	1810	1838	1894	1945	1978	2010	2045	2076	2108	2140	2170
.ACB	2200	2246	2328	2365										
	298#	298	299#	299	410#	410	411#	411	422#	422	423#	423	457#	457
	458#	458	499#	499	511#	511	512#	512	592#	592	593#	593	637#	637
	638#	638	885#	885	886#	886	1001#	1001	1002#	1002	1033#	1033	1034#	1034
	1081#	1081	1082#	1082	1183#	1183	1184#	1184	1220#	1220	1221#	1221	1237#	1237
	1238#	1238	1271#	1271	1272#	1272	1291#	1291	1292#	1292	1318#	1318	1319#	1319
	1351#	1351	1352#	1352	1403#	1403	1404#	1404	1457#	1457	1458#	1458	1482#	1482
	1510#	1510	1511#	1511	1520#	1520	1521#	1521	1573#	1573	1574#	1574	1612#	1612
	1613#	1613	1657#	1657	1658#	1658	1703#	1703	1704#	1704	1739#	1739	1740#	1740
	1778#	1778	1779#	1779	1810#	1810	1811#	1811	1838#	1838	1839#	1839	1894#	1894
	1895#	1895	1945#	1945	1946#	1946	1978#	1978	1979#	1979	2010#	2010	2011#	2011
	2045#	2045	2046#	2046	2076#	2076	2077#	2077	2108#	2108	2109#	2109	2140#	2140
	2141#	2141	2170#	2170	2171#	2171	2200#	2200	2201#	2201	2245#	2245	2328#	2328
.ACM	2329#	2329	2365#	2365	2366#	2366								
	298#	298	299	410#	410	411	422#	422	423	457#	457	458	499#	499
	500	511#	511	512	592#	592	593	637#	637	638	885#	885	886	1001#
	1001	1002	1033#	1033	1034	1081#	1081	1082	1183#	1183	1184	1220#	1220	1221
	1237#	1237	1238	1271#	1271	1272	1291#	1291	1292	1318#	1318	1319	1351#	1351
	1352	1403#	1403	1404	1457#	1457	1458	1482#	1482	1483	1510#	1510	1511	1520#
	1520	1521	1573#	1573	1574	1612#	1612	1613	1657#	1657	1658	1703#	1703	1704
	1739#	1739	1740	1778#	1778	1779	1810#	1810	1811	1838#	1838	1839	1894#	1894
	1895	1945#	1945	1946	1978#	1978	1979	2010#	2010	2011	2045#	2045	2046	2076#
	2076	2077	2108#	2108	2109	2140#	2140	2141	2170#	2170	2171	2200#	2200	2201
	2245#	2245	2246	2328#	2328	2329	2365#	2365	2366					
.CHCRT	348													
.CHLFD	342	350												
.FHSLF	461	307	1310	1325	1380	1394	2213							
.LBCON	1897													
.LBCSP	2079													
.LBDIS	1981													
.LBEVT	1948													
.LBGDE	2173													
.LBQRM	2048													
.LBRCD	1841													
.LBRMG	2143													
.LBSMG	2013													
.LBSTS	2111													
.NVR	298#	298	299	299#	410#	410	411	411#	422#	422	423	423#	457#	457
	458	458#	499#	499	500	511#	511	512	512#	592#	592	593	593#	637#
	637	638	638#	885#	885	886	886#	1001#	1001	1002	1002#	1033#	1033	1034
	1034#	1081#	1081	1082	1082#	1183#	1183	1184	1184#	1220#	1220	1221	1221#	1237#
	1237	1238	1238#	1271#	1271	1272	1272#	1291#	1291	1292	1292#	1318#	1318	1319
	1319#	1351#	1351	1352	1352#	1403#	1403	1404	1404#	1457#	1457	1458	1458#	1482#
	1483	1510#	1510	1511	1511#	1520#	1520	1521	1521#	1573#	1573	1574	1574#	1612#
	1612	1613	1613#	1657#	1657	1658	1658#	1703#	1703	1704	1704#	1739#	1739	1740
	1740#	1778#	1778	1779	1779#	1810#	1810	1811	1811#	1838#	1838	1839	1839#	1894#
	1894	1895	1895#	1945#	1945	1946	1946#	1978#	1978	1979	1979#	2010#	2010	2011

SEQ 0167

[illegible]

.SSCON	1912	2273		
.SSCSP	2085	2274		
.SSDIS	1988	2275		
.SSEVT	1953	2276		
.SSGDE	2178	2277		
.SSQRM	2056	2278		
.SSRCD	1847	2279		
.SSRMG	2148	2280		
.SSSMG	2021	2281		
.SSSTS	2116	2231	2282	
.TICCC	185	1323	1377	
.TICTI	411	1311	1321	1375
.TIMEL	461			

SEQ 7169

AIC%	1327														
ASSCHN	233#	245	246	247	248	249	250	251							
ATI%	1322	1324													
CIS%	1384														
CLRBFI	425														
DEBRK%	289	408	438	477	482	487	492								
DEBRK.	294	420	443	497											
DIC%	1332														
DIR%	1383														
DTI%	412	1376	1378												
ERJMP	2211	2345	2376												
GETER%	2214														
GETTAB	1246	1253													
GLOB	139														
HIBER	204														
JUMPF	139	326													
JUMPT	139	305													
LSTOF.	138	181	225	299	301	368	411	423	429	449	451	458	466	471	
	477	482	507	512	514	538	551	578	583	590	593	598	600	602	
	604	606	608	613	618	623	638	640	655	669	671	681	708	715	
	722	751	762	765	769	789	803	806	810	818	835	839	844	849	
	855	866	886	888	906	921	924	928	936	958	971	974	978	984	
	1002	1004	1011	1015	1034	1037	1047	1050	1054	1060	1082	1090	1098	1103	
	1109	1115	1122	1126	1132	1135	1161	1184	1190	1206	1209	1221	1238	1272	
	1292	1319	1335	1338	1341	1344	1347	1352	1361	1364	1367	1370	1373	1404	
	1458	1459	1477	1511	1521	1557	1574	1613	1658	1704	1740	1779	1811	1820	
	1839	1853	1895	1920	1946	1959	1979	1994	2011	2027	2046	2062	2077	2091	
	2109	2122	2141	2154	2171	2184	2201	2250	2258	2268	2329	2339	2348	2366	
	2373	2379													
LSTON.	299	411	423	458	500	512	593	638	886	1002	1034	1082	1184	1221	
	1238	1272	1292	1319	1352	1404	1458	1483	1511	1521	1574	1613	1658	1704	
	1740	1779	1811	1839	1895	1946	1979	2011	2046	2077	2109	2141	2171	2201	
	2246	2329	2366												
MOVX	468	1311	1321	1323	1326	1381	2233								
NAME	106														
NOP	220	709													
OUTSTR	424														
PISYS.	427	1333	1336	1339	1342	1345	1359	1362	1365	1368	1371				
PITMR.	469														
PJRST	139														
PROLOG	138														
RFMOD%	1301														
RTIW%	1308														
SCS%	2210														
SCS.	2224														
SFMOD%	1305	1391													
SIBEX	397														
SKPINC	401														
STIW%	1312	1396													
STPAR%	1392														
SYERR%	2344	2375													
TIME%	1240														
TIMER%	464														

SEQ 0170

TXNE
WAIT%
\$CALL

502
208
155
293
449
539
639
747
809
920
1005
1089
1223
1366
1534
1919
2153
2139#
583
1015
1445
1860
2268

504
156
300
450
550
641
748
812
923
1008
1097
1226
1369
1538
1954
2179
139
590
1064
1470
1921
2331

162
304
465
577
654
750
817
927
1010
1102
1244
1372
1556
1958
2183
225
608
1162
1477
1960
2346

167
325
470
582
668
752
834
930
1013
1108
1277
1458
1585
1989
2245
310
613
1210
1491
1994
2348

169
347
476
589
670
756
838
935
1014
1114
1280
1463
1624
1993
2249
314
618
1233
1515
2027
2368

180
352
481
597
680
761
843
957
1034
1121
1296
1465
1669
2022
2257
415
623
1241
1557
2062
2377

182
356
486
599
685
764
848
962
1036
1125
1334
1475
1715
2026
2267
432
626
1250
1588
2091
2379

190
367
491
601
698
768
854
966
1038
1131
1337
1476
1749
2057
2335
466
629
1259
1627
2123

193
407
496
603
701
769
865
970
1042
1134
1340
1482
1790
2061
2338
471
733
1287
1672
2155

219
419
499
605
707
788
887
973
1046
1160
1343
1512
1816
2086
2347
501
771
1298
1718
2184

221
428
506
607
708
793
903
977
1049
1189
1346
1514
1819
2090
2369
507
867
1314
1752
2216

222
437
513
612
714
797
905
979
1053
1205
1352
1523
1848
2117
2372
552
941
1373
1793
2229

224
442
515
617
715
802
909
983
1055
1208
1360
1527
1852
2121
2378
573
989
1399
1821
2232

288
448
537
622
721
805
915
1003
1059
1209
1363
1531
1913
2149
578
1011
1407
1853
2237

SEQ 0171

\$RET

\$RETF
\$RETIF
\$RETI
\$RETT
\$SAVE

\$TEXT

..POP

..PUSH

.NV1

.NV2

298
1237
1739
2245
179
550
653
809
957
1096
1340
1958
2378

410
1271
1778
2328
223
577
668
816
970
1101
1343
1993

422
1291
1810
2365
300
581
669
834
973
1108
1346
2026

457
1318
1838
367
588
679
838
977
1114
1360
2061

499
1351
1894
428
597
707
843
983
1121
1363
2090

511
1403
1945
448
599
714
848
1003
1125
1366
2121

592
1457
1978
450
601
721
854
1010
1131
1369
2153

637
1482
2010
465
603
750
864
1014
1134
1372
2183

885
1510
2045
470
605
761
887
1036
1158
1458
2249

1001
1520
2076
476
607
764
905
1046
1189
1476
2257

1033
1573
2108
481
612
768
920
1049
1204
1548
2266

1081
1612
2140
506
616
788
923
1053
1208
1819
2337

1183
1657
2170
513
622
802
927
1059
1334
1852
2347

1220
1703
2200
537
639
805
934
1089
1337
1919
2371

299
1272
1811
299
1272
1811
299#
886#
1272#
1521#
1811#
2077#
299#

411
1292
1839
411
1292
1839
299
886
1272
1521
1811
2077
299

423
1319
1895
423
1319
1895
411#
1002#
1292#
1574#
1839#
2109#
411#

458
1352
1946
458
1352
1946
411
1002
1292
1574
1839
2109
411

512
1404
1979
512
1404
1979
423#
1034#
1319#
1613#
1895#
2141#
423#

593
1458
2011
593
1458
2011
423
1034
1319
1613
1895
2141
423

638
1511
2046
638
1511
2046
458#
1082#
1352#
1658#
1946#
2171#
458#

886
1521
2077
886
1521
2077
458
1082
1352
1658
1946
2171
458

1002
1574
2109
1002
1574
2109
512#
1184#
1404#
1704#
1979#
2201#
512#

1034
1613
2141
1034
1613
2141
512
1184
1404
1704
1979
2201
512

1082
1658
2171
1082
1658
2171
593#
1221#
1458#
1740#
2011#
2329#
638#

1184
1704
2201
1184
1704
2201
593
1221
1458
1740
2011
2329
638

1221
1740
2329
1221
1740
2329
638#
1238#
1511#
1779#
2046#
2366#
886#

1238
1779
2366
1238
1779
2366
638
1238
1511
1779
2046
2366
886

	1002#	1002	1082#	1082	1184#	1184	1221#	1221	1272#	1272	1292#	1292	1319#	1319
	1352#	1352	1521#	1521	1574#	1574	1613#	1613	1658#	1658	1704#	1704	1740#	1740
	1779#	1779	1811#	1811	1839#	1839	1895#	1895	1946#	1946	1979#	1979	2011#	2011
	2046#	2046	2077#	2077	2109#	2109	2141#	2141	2171#	2171	2201#	2201	2329#	2329
	2366#	2366												
.NV3	458#	458	638#	638	886#	886	1082#	1082	1184#	1184	1292#	1292	1521#	1521
	1839#	1839	2011#	2011	2077#	2077	2201#	2201						
.NV4	638#	638	1082#	1082	1521#	1521	2011#	2011						
.NV5	638#	638	1521#	1521										
.NV6	638#	638	1521#	1521										

SEQ 0172

